Local Principal Curves

Jochen Einbeck[*]
Department of Mathematics, National University of Ireland,
Galway, Ireland

Gerhard Tutz[†]
Institut für Statistik, Ludwig-Maximilians-Universität, Akademiestr. 1,
80799 München, Germany

Ludger Evers[‡]
Department of Statistics, University of Oxford, 1 South Parks Road,
Oxford OX1 3TG, UK.

12th June 2007

Abstract

Principal components are a well established tool in dimension reduction. The extension to principal curves allows for general smooth curves which pass through the middle of a multidimensional data cloud. In this paper local principal curves are introduced, which are based on the localization of principal component analysis. The proposed algorithm is able to identify closed curves as well as multiple curves which may or may not be connected. For the evaluation of the performance of principal curves as tool for data reduction a measure of coverage is suggested. By use of simulated and real data sets the approach is compared to various alternative concepts of principal curves.

Key Words: Local smoothing, mean shift, principal components, principal curves.

[*]jochen.einbeck@nuigalway.ie
[†]tutz@stat.uni-muenchen.de
[‡]evers@stats.ox.ac.uk

1 Introduction

The classical problem of how to find the best curve passing through data points $(x_i, y_i), i = 1, \dots, n$ can be handled in two fundamentally different ways. Let us regard the data points as realizations of i.i.d. random variables (X_i, Y_i) drawn from a population (X, Y). A common approach is to regard X as an explanatory variable for the dependent variable Y. This concept is used when the focus is on regression and is especially useful when the objective is the prediction of the dependent variable from observations x_i. Thereby X and Y have an asymmetric relationship and cannot be interchanged without affecting the results.

In contrast, X and Y may be regarded as symmetric, thus it is not assumed that one variable depends on the other one. These approaches are useful when the focus is on dimension reduction or simply description of the data. Representants are methods like the ACE algorithm, canonical correlation or principal component analysis. Linear principal components, introduced by Pearson (1901), are a common tool in multivariate analysis, applied for example in feature extraction or dimension reduction. Jolliffe (2002) gave an extensive overview on properties and applications of principal components. Nonlinear principal components have been developed by Schölkopf & Smola (1998) and successfully employed in pattern recognition.

A natural extension of principal components are principal curves, which are descriptively defined as one-dimensional smooth curves that pass through the "middle" of a $d-$dimensional data set. Although this concept is intuitively clear, there is much flexibility in how to define the "middle" of a distribution or data cloud. Hastie & Stuetzle (1989) (hereafter HS), who did the groundbreaking work on principal curves, use the concept of self-consistency (Tharpey & Flury, 1996), meaning that each point of the principal curve is the average over all points that project there. A variety of other definitions of principal curves have been given subsequently by Tibshirani (1992), Kégl, Krzyzak, Linder & Zeger (2000) (hereafter KKLZ), and more recently Delicado (2001), which differ essentially in how the "middle" of the distribution is found.

The existing principal curve algorithms can be divided into two families: Firstly, there is one family of algorithms based on "top-down"- strategies. These algorithms start with a straight line, which is mostly the first principal component of the data set, and try to dwell out this line or concatenate other lines to the initial line until the resulting curve passes satisfactorily through the middle of the data. However, the dependence on an initial line leads to some technical problems and a lack of flexibility. For instance, principal curves according to HS are often strongly biased, they exclude by construction the handling of crossings or branched curves, and they are not able to handle closed curves. Banfield & Raftery (1992) provide a bias corrected version of the HS algorithm which solves the latter problem. Chang & Ghosh (1998) combine the algorithms of HS and Banfield/Raftery and show that this yields a smooth and unbiased principal curve, at least for simple data situations. Tibshirani's theoretically attractive approach seems to have similar problems as HS, and in addition it seems to be not flexible enough to recover curves with high curvature. These difficulties have been solved by Verbeek, Vlassis & Kröse (2001), but at the expense of an apparently wiggly principal curve, since polygonal lines are connected in a somehow unsmooth manner. KKLZ also work with polygonal lines and obtain with high computational effort a smooth and flexible principal curve, which only fails for very complicated data structures. None of these algorithms seems to be able to handle curves which consist of multiple or disconnected parts, at least not directly without some modifications or improvements. Recently, Kégl & Krzyzak (2002) provided a promising algorithm to obtain *principal graphs*, i.e. multiple connected piecewise linear curves, in the context of skeletonization of hand-written characters.

The aim to handle complex data structures like spirals or branched curves motivates a concept that differs from the above mentioned ones. Instead of starting with a *global* initial line, it often seems more appropriate to construct the principal curve in a "bottom-up" manner by considering in every step only data in a *local* neighborhood of the currently considered point. Recently, Delicado (2001) proposed the first principal curve approach which can be assigned to this family.

Let X be a d-dimensional random vector and $X_i \in \mathbb{R}^d, i = 1, \ldots, n$ denote an iid. sample from X, where $X_i = (X_{i1}, \ldots, X_{id})$. For each point x, Delicado considers the hyperplane $H(x, b)$ which contains x and is orthogonal to the vector b. The set of vectors $b^*(x)$ minimizing the total variance $\phi(x, b) = TV(X|X \in H(x, b))$ defines a function $\mu^*(x) = E(X|X \in H(x, b^*(x)))$. Principal oriented points (POPs) are introduced as fix points of the function $\mu^*(\cdot)$. For a suitable interval $I \in \mathbb{R}$, α is called a principal curve of oriented points (PCOP) if $\{\alpha(s)|s \in I\}$ is a subset of the fix point set of μ^*. Delicado shows that POPs exist, and that in the case where $b^*(x)$ is unique, for each POP exists a PCOP passing through it. Since the hyperplanes H are sets of measure zero, it is necessary to employ a kind of smoothing for calculating the conditional expectation on the hyperplane. This is achieved by projecting all data points on $H(x, b)$, obtaining points X_i^H, and assigning weights

$$\bar{w}_i = \bar{w}(|(X_i - x)^T b|), \tag{1}$$

where $\bar{w}$ is a decreasing positive function, e.g. $\bar{w}(\cdot) = K(\cdot/h)$, with a kernel function K. Let $\tilde{\mu}(x, b)$ denote the weighted expectation of the X_i^H with weights $\bar{w}_i$. Now $\mu^*(x)$ is approximated by $\tilde{\mu}^*(x) = \tilde{\mu}(x, \tilde{b}^*(x))$, where $\tilde{b}^*(x)$ (and hence H) is constructed such that the variance of the projected sample, weighted with $\bar{w}_i$, is minimized. Localization enters here twofold. Firstly, by using the weights (1), points near the hyperplane are upweighted. Secondly, a cluster analysis is performed on the hyperplane, and only data in the local cluster are considered for averaging. The algorithm searches the fix point set of $\tilde{\mu}^*(\cdot)$ as follows. Repeatedly, choose a point randomly from the sample $X_1, \ldots, X_n$ and call it $x_{(0)}$. Then iterate $x_{(\ell)} = \tilde{\mu}^*(x_{(\ell-1)})$ until convergence. In this manner a finite set of POPs is obtained. However, no fix point theorem guarantees convergence of this algorithm, although Delicado reports quick convergence for some real data sets. In order to obtain a PCOP from a set of POPs, Delicado proposes an idea which we will further exploit. Assume an POP x_1 has been calculated. From the set of principal directions $\tilde{b}^*(x_1)$, choose one vector b_1. Now take a step of length ∂ from x_1 into the direction of b_1, i.e.

$$x_2^0 = x_1 + \partial b_1, \tag{2}$$

where ∂ is previously fixed. The point x_2^0 serves as a new starting point for a new iterating process, leading to a new point x_2 of the principal curve. This is repeated k times until no points X_i can be considered to be near the hyperplane $H(x_k^0, b_k)$. Then return to (x_1, b_1) and complete the principal curve in direction of $-b_1$. Afterwards move on to another of the previously chosen POPs and continue analogously.

Delicado's concept is mathematically elegant and theoretically well elaborated. Delicado & Huerta (2003) show that it works fine even for some complicated data structures, and provide a method for selection of the parameter h. One might consider it as a drawback that the concept is mathematically demanding and computationally extensive, since a large number of cluster analyses has to be run. In this paper, we introduce a concept similar to that of Delicado. However, we replace the fix points of $\tilde{\mu}^*$ by local centers of mass, and replace the principal direction b_1 by a local principal component. We call the resulting curve, which consists of a series of local centers of mass, *local principal curve.* The crucial advantage of this concept is that calculation of the local principal component results from a simple eigenvalue problem, making computation extremely fast and easy.

In addition, we introduce the notion of *coverage*, which evaluates the performance of the principal curve approximation and is a helpful tool to compare the performance of different principal curve algorithms. The price to be paid for the computational advantages of the concept is that in contrast to Delicado's approach there is no statistical model and consequently it is hard to derive theoretical results. However, in Section 5 we show that our algorithm can be seen as a simple and fast approximation to Delicado's algorithm. The algorithm to construct local principal curves, hereafter LPC, will be presented in the following section.

2 The LPC algorithm

Assume a d-dimensional data cloud $X_i \in \mathbb{R}^d, i = 1, \ldots, n$, where $X_i = (X_{i1}, \ldots, X_{id})$. We try to find a smooth curve which passes through the "middle" of the data cloud. The curve will be calculated by means of a series of local centers of mass of the data, according to the following strategy:

1. Choose a suitable starting point $x_{(0)}$. Set $x = x_{(0)}$.
2. Calculate the local center of mass μ^x around x.
3. Perform a principal component analysis locally at x.
4. Find the new value x by following the first local principal component γ^x starting at μ^x.
5. Repeat steps 2 to 4 until μ^x remains (approximately) constant.

The series of the μ^x determines the local principal curve. In the sequel we will explain these steps in detail:

1. Selection of the starting point

In principle, every point $x_{(0)} \in \mathbb{R}^d$ which is in or close to the data cloud can be chosen as starting point. There are two ideas which suggest themselves:

- Based on a density estimate the point with the highest density $x_{(0)} = max_{x \in \mathbb{R}} \hat{f}(x)$ is chosen.
- A point $x_{(0)} = X_i$ is chosen at random from the set of observations.

The advantage of the density method is that one can be quite sure not to start in a blind alley, whereas a randomly chosen point could be an outlier far from the data cloud which stops the algorithm already in the first loop. However, in this case it is easy to draw another starting point, and the computational costs of the second approach are much lower. Moreover, for the handling of crossings a randomly chosen starting point is even superior to a high density point.

2. Calculating the local center of mass

Let H be a bandwidth matrix and $K_H(\cdot)$ a $d-$ dimensional kernel function. Given that all components of X are measured on the same scale, we set $H = \{h^2 \cdot I : h > 0\}$, with I denoting the d-dimensional identity matrix. For a detailed description of multivariate kernels and bandwidth matrices see Wand & Jones (1993). For some remarks on the selection of h, see Section 6. The local center of mass around x is given by

$$\mu(x) = \frac{\sum_{i=1}^n K_H(X_i - x) X_i}{\sum_{i=1}^n K_H(X_i - x).} \tag{3}$$

Comaniciu & Meer (2002) studied the properties of the *mean shift*, which is given by $\mu(x) - x$, and investigated the relation of $\mu(x)$ to the Nadaraya-Watson estimator. For ease of notation, we will use the abbreviation $\mu^x = \mu(x)$, and denote the j-th component of $\mu(x)$ by μ_j^x.

3. Calculating the local principal component

Let $\Sigma^x = (\sigma_{jk}^x)$ denote the local covariance matrix of x, whose (j,k)-th entry $(1 \le j, k \le d)$ is given by

$$\sigma_{jk}^x = \sum_{i=1}^n w_i (X_{ij} - \mu_j^x)(X_{ik} - \mu_k^x) \tag{4}$$

with weights $w_i = K_H(X_i - x) / \sum_{i=1}^n K_H(X_i - x)$, and H as in 2. Let γ^x be the first eigenvector of Σ^x. Then γ^x is the first column of the loadings matrix Γ^x from the eigen decomposition $(\Gamma^x)^T \Sigma^x \Gamma^x = \Lambda^x$, where $\Lambda^x = \text{diag}(\lambda_1^x, \ldots, \lambda_d^x)$ is a diagonal matrix containing the ordered eigenvalues of Σ^x, with $\lambda_1^x \geq \ldots \geq \lambda_d^x$.

It should be noted that the denotation "local principal components" has been previously used for linear principal components which are localized in clusters (Skarbek, 1996; Kambhatla & Leen, 1997) or based on contiguity relations (Aluja-Banet & Nonell-Torrent, 1991). Here localization refers to the use of kernel functions that define a neighborhood.

4. Obtaining an updated value

The local principal component line v^x can now be parameterized by

$$v^x(t) = \mu^x + t\gamma^x \quad (t \in \mathbb{R}), \tag{5}$$

and an updated value of x is obtained by setting

$$x := \mu^x + t_0\gamma^x, \tag{6}$$

similar as in (2) of Delicado's algorithm. A suitable value of t_0 thereby has to be chosen beforehand. In all examples in this paper we employ the simple rule $t_0 = h$.

5. Stop when μ^x remains constant

When the margin of the data cloud is reached, the algorithm naturally gets stuck and produces approximately constant values of μ^x. One might also stop before this state occurs, e.g. when the difference between the previous and the current center of mass falls below a certain threshold.

The mechanism is demonstrated in Fig. 1. The starting point $x_{(0)}$ is denoted by 0. The radius of the circle equals the bandwidth $h = 0.2$. Calculating the local center of mass around 0 yields the nearby point m. Moving along the first principal component with $t_0 = 0.2$ leads to the new point x denoted by "1", and so on. The series of m's represents the local principal curve. Note that the algorithm is based on finding an equilibration between opposing tendencies: On the one hand, the local principal components are oversteering, i.e. tending "outside" to the concave side of the curvature of the data cloud. On the other hand, the calculation of the local center of mass smoothes the data towards the interior and thus in the opposite direction, effecting a small bias of the local principal curve.

We remark that principal curves by HS show a similar behavior: Theoretically, self-consistent principal curves are biased outwards, as explained by Tibshirani (1992). Practically, they are often biased inwards due to smoothing as demonstrated by Chang & Ghosh (1998). The often observed large bias of the HS principal curves is due to another reason: An initial unsuitable assignment of projection indices cannot be corrected in the further run of the algorithm.

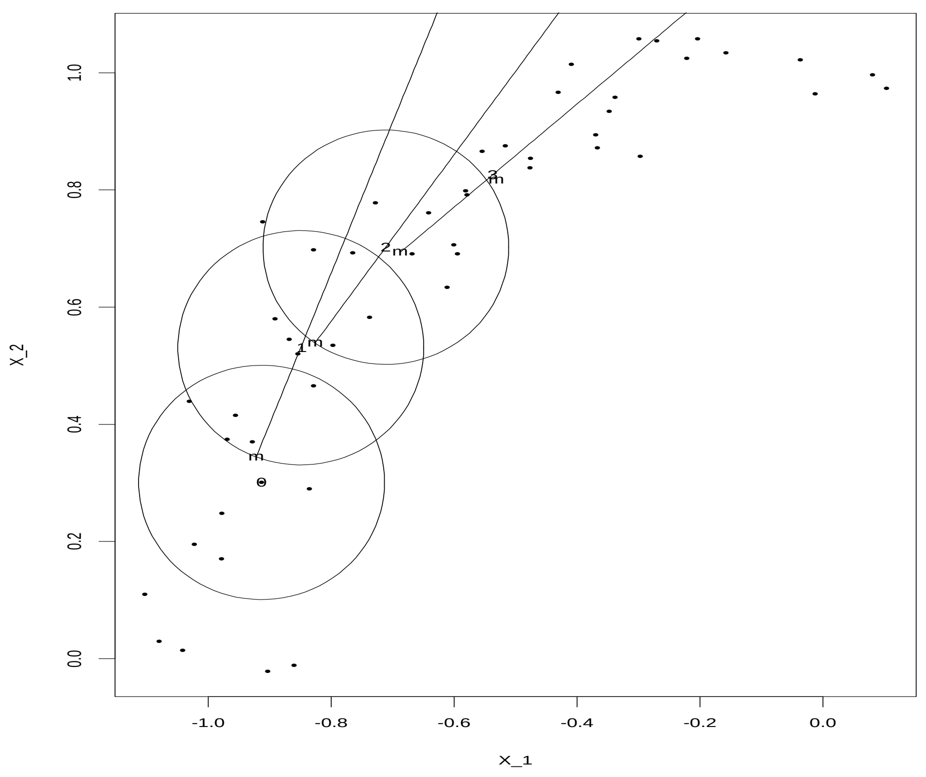

Figure 1: Demonstration of the local principal curve algorithm.

3 Technical details

3.1 Maintaining the direction

If the orientation of the eigenvector changes from one step to another, the algorithm dangles between two points and will never escape. Therefore one should check in every step that the local eigenvector has the same direction as in the previous step. This can be done by simply calculating the angle $\alpha^x_{(i)}$ between the eigenvectors $\gamma^x_{(i-1)}$ and $\gamma^x_{(i)}$ belonging to the $(i-1)$-th resp. i-th step , which is given by

$$\cos(\alpha^x_{(i)}) = \gamma^x_{(i-1)} \circ \gamma^x_{(i)},$$

where $\circ$ denotes the scalar product. If $\cos(\alpha^x_{(i)}) < 0$, set $\gamma^x_{(i)} := -\gamma^x_{(i)}$, and continue the algorithm as usual. This “signum flipping” has been applied in the step from “2” to “3” in Figure 1.

3.2 Running backwards from $x_{(0)}$

When one starts at a point $x_{(0)}$ and moves by means of local principal components to one "end" of the cloud, one has neglected that the opposite direction is equally adequate. Similar as in Delicado (2001), an additional step has to be added to the algorithm in practice:

6. For the starting direction $-\gamma^x_{(0)} := -\gamma^{x(0)}$, perform steps 4 and 5.

This step can be omitted when the data describe a closed curve, e.g. a spiral or an ellipse.

3.3 Angle penalization

If the data cloud locally forms crossings, at each crossing there are three possibilities for the local principal curve where to move on. One often prefers that the curve passes straight on at each crossing, and does not turn arbitrarily to the left or right. In order to achieve this effect, we recommend to perform an angle penalization in addition to the signum flipping in each step of the algorithm. This might be done as follows:

Let k be a positive number. For the angle $\alpha^x_{(i)}$, set

$$a^x_{(i)} := |\cos(\alpha^x_{(i)})|^k$$

and correct the eigenvectors according to

$$\gamma^x_{(i)} := a^x_{(i)} \cdot \gamma^x_{(i)} + (1 - a^x_{(i)}) \cdot \gamma^x_{(i-1)}$$

Thus, the higher the value of k, the more the curve is forced to move straight on. We recommend to set set $k = 1$ or 2. For higher values of k the local principal curve looses too much flexibility. A similar angle penalty is used in the implementation of KKLZ's algorithm. We state explicitly that we do not introduce the angle penalty to improve the smoothness of the curve - the local principal curve, computed with a suitable bandwidth, is already sufficiently smooth. The motivation is mainly to handle data with crossings (see Section 4.2).

3.4 Multiple initializations

Assume that the data cloud consists of several branches, which might or might not be connected. Then one single local principal curve will fail to describe the whole data set.

In our approach this problem can be solved by initializing more than once, i.e. we choose subsequently a series of starting points, so that finally at least one starting point is situated on each branch, and run the algorithm for each starting point. Following this procedure the whole data cloud will be covered by the local principal curve. The starting points can be imposed by hand on each of the branches, or, if this is not possible or too cumbersome, they might be chosen at random. If one has for example two disconnected branches of the data cloud, which contain more or less the same amount of data, then the application of four randomly chosen starting points already effects that with a probability of 93.75% at least one starting point is in each cloud. For an arbitrary number of branches, Borel-Cantelli's Lemma tells us that with the number of starting points increasing to infinity, each branch is visited with probability 1. In practice this technique proves to work satisfactorily, even for a high number of branches. To conclude, for a set of starting points S_0, we add a 7th step to the algorithm:

7. If $S_0 \neq \emptyset$, choose (without replacement) a new starting point $x_{(0)} \in S_0$ and start again with step 1.

It should be noted that our algorithm is deterministic given the starting points, but yields different principal curves for different starting points. However, since in each case the local centers of mass of the same data are calculated, differences of principal curves on the same branch are usually negligible.

4 Applications

4.1 2-dimensional data

Firstly, we compare the results of our algorithm with some simulated standard examples, similar to those examined by Kégl, Krzyzak, Linder & Zeger (2000) and Delicado & Huerta (2003). (In this and the following examples, the curves from KKLZ are obtained via the Principal Curves Java program from Balázs Kégl, available at `http://www.iro.umontreal.ca/~kegl/research/pcurves/`. The HS curves were obtained by Hastie's Splus function `http://lib.stat.cmu.edu/S/principal.curve`. Principal curves according to Delicado are computed with a C++ program, which is available at `http://www-eio.upc.es/~delicado/PCOP`.). We consider eight different scenarios: Firstly, $n = 100$ data points are generated by means of an underlying circle of radius $r = 1$, contaminated with small noise

($\sigma = 0.01$) and with large noise ($\sigma = 0.2$), respectively. Further, we consider four types of spirals: a simple spiral with small ($\sigma = 0.01$) and large noise ($\sigma = 0.06$), a complex spiral with small ($\sigma = 0.01$) and large noise ($\sigma = 0.05$), each for $n = 1000$. Finally, we investigate a zigzag pattern with small ($\sigma = 0.008$) and large noise ($\sigma = 0.05$), where in both cases $n = 400$ data points were generated. The given values of σ have to be understood as Gaussian noise which is independently imposed on the x - and y- coordinate of the corresponding underlying curves.

The results obtained by applying the "top-down" strategies from HS and KKLZ are presented in Fig. 2, and the principal curves constructed in a "bottom-up" manner (Delicado, LPC) are shown in Fig. 3. Looking at the data with moderate noise on the left side of Fig. 2 and 3, respectively, one notices that most algorithms yield satisfactory results, except the HS algorithm. Delicado's algorithm fails to handle the complex spiral. The curve obtained by LPC is nearly indistinguishable from the true curve in all four cases, except the peaks of the zigzag curve. Regarding the noisy data, one sees that the circle is reconstructed quite differently by all four algorithms, whereby only LPC and Delicado lead to a closed curve. HS and Delicado have serious problems with the small noisy spiral. The result of KKLZ seems to be perfect in this case. The result of LPC is quite good, but there are two artificial lines connecting different parts of the spiral. HS, Delicado, and KKLZ fail for the complex noisy spiral. The local principal curve succeeds to follow the spiral, although once again again some artificial loops appear. The noisy zigzag data are fitted best by KKLZ and worst by HS. Delicado and LPC perform very similarly for these data. In Section 6 we will evaluate the performance of the principal curves quantitatively.

Finally, we consider real data recorded by the Office of Remote Sensing for Earth Resources, Pennsylvania State University, which show the location of floodplains in Beaver County, PA, USA, 1996 (Fig. 4). For analyzing the data, we digitalized the map to a grid of $106 \times 70 = 7420$ digits. Fig. 5 shows the result of a run of the LPC algorithm using the digitalized floodplain data. We used 50 initializations and a bandwidth $h = 1.5$. The principal curve uncovers nicely the principal courses of the floodplains. Taking a look at maps from Beaver county, we see that our principal curve reconstructs the underlying rivers resp. valleys in this district (The corresponding maps are available at PASDA - Pennsylvania Spatial Data Access, `www.pasda.psu.edu`, and can be regarded with the ArcExplorerWeb at `www.esri.com/software/arcexplorer`). Note that a quite big cluster in the central bottom is not covered - this simply occurs because none of the randomly chosen starting points is situated there, and this isolated cluster cannot be reached by an

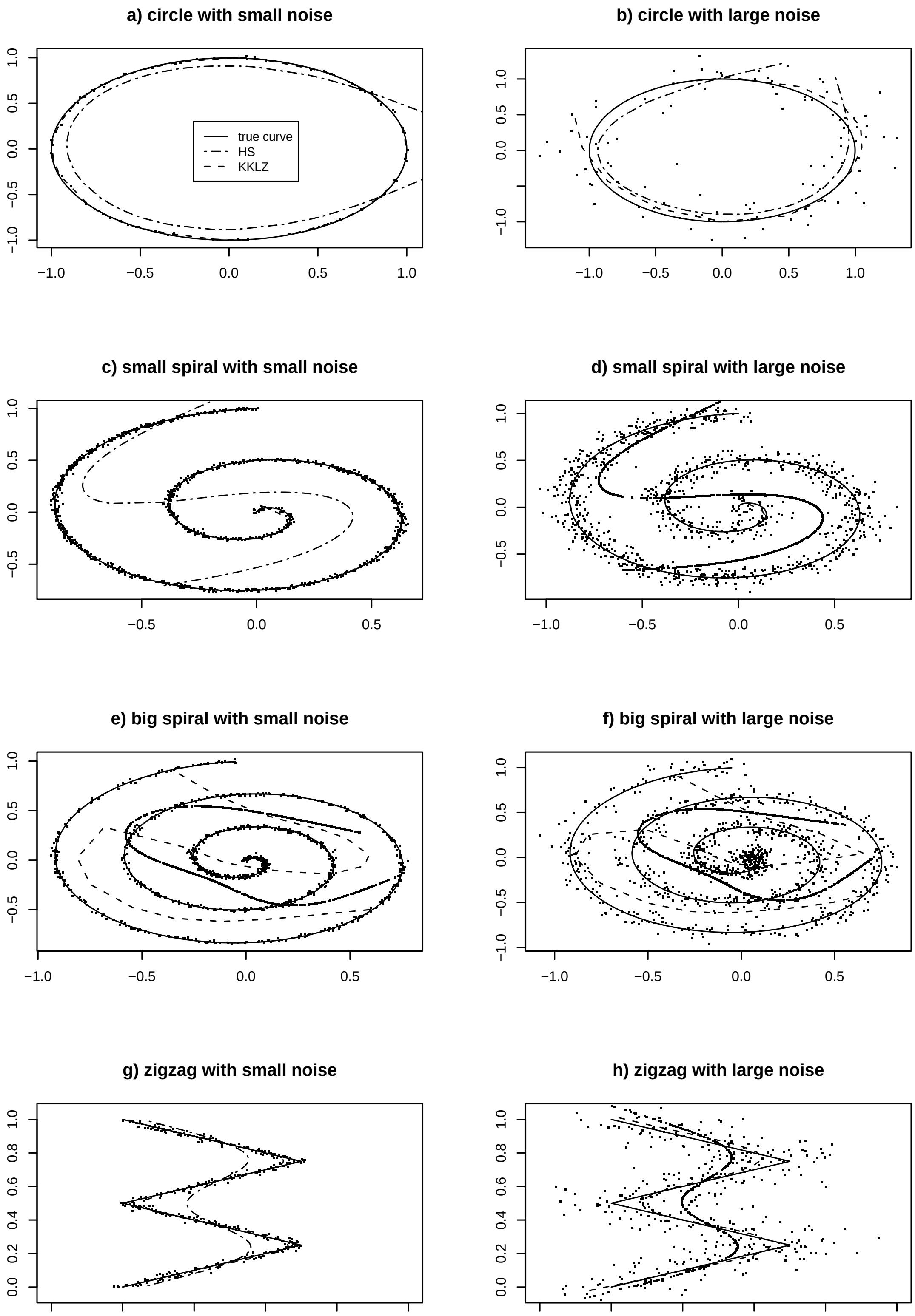

Figure 2: Principal curves according to HS and KKLZ for various data situations.

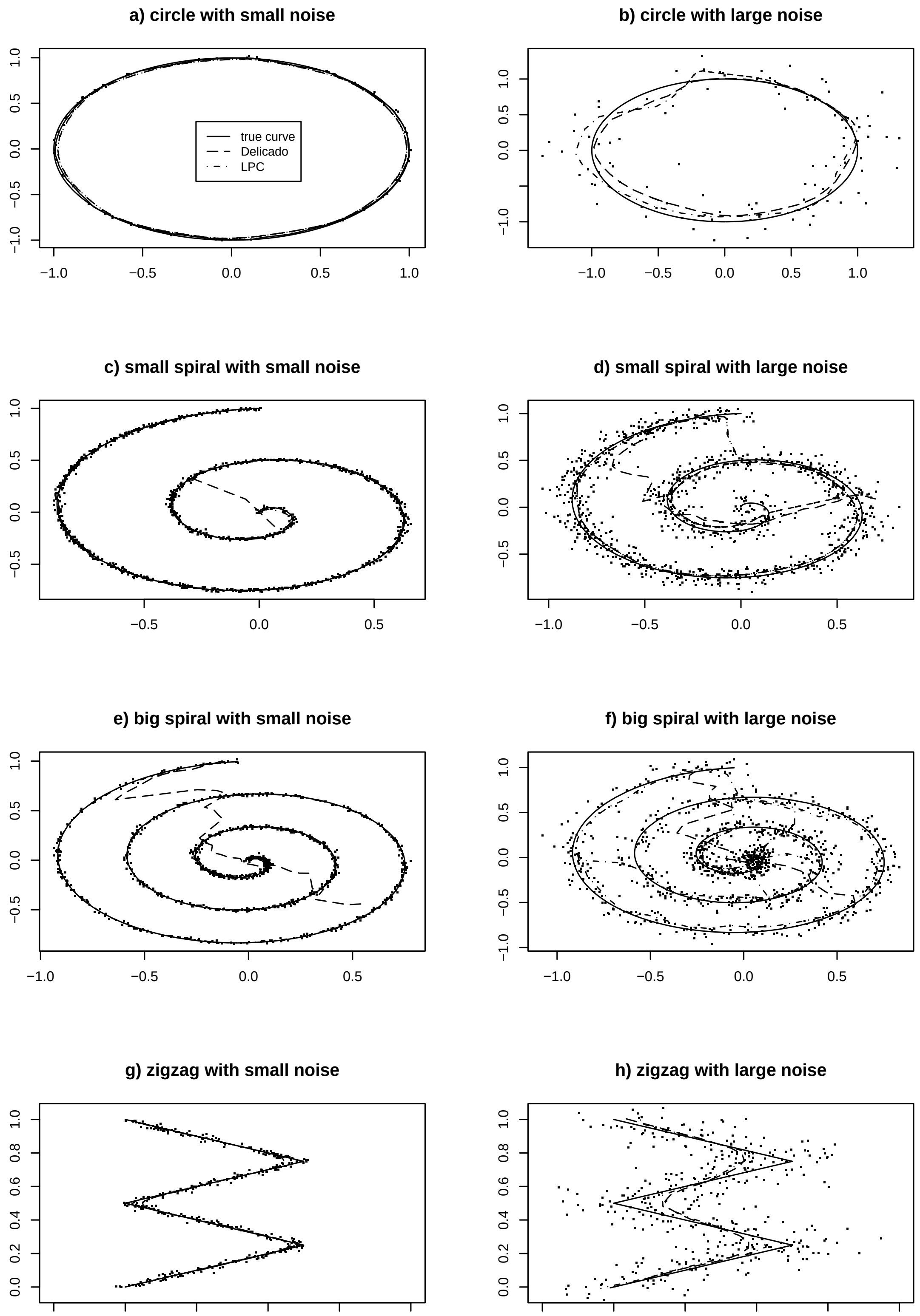

Figure 3: Local principal curves compared with Delicado's PCOPs.

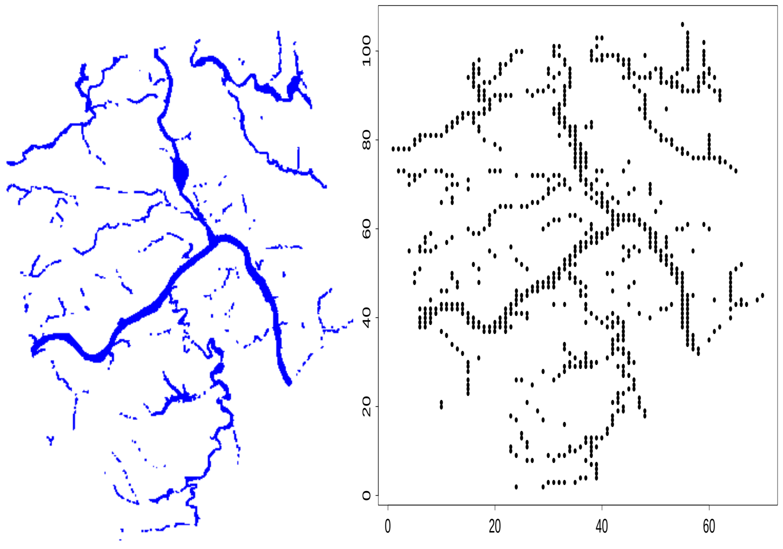

Figure 4: Floodplains in Beaver County, PA. (left: original, right: digitalized).

external principal curve. More initializations would be necessary to solve this.

4.2 3-dimensional data

We now consider a data set included in the S-Plus software package, namely the "radial velocity of galaxy NGC7531". This data frame, recorded by Buta (1987), contains the radial velocity of 323 points of that spiral galaxy covering about 200 arc seconds in north-south and 135 arc seconds in east-west direction in the celestial sphere. All of the measurements lie within seven slots crossing the origin. The x- and y-coordinate describe the east-west resp. north-south coordinate, and the z-coordinate is the radial velocity measured in km/sec. For simplicity, we only consider the first 61 data points of the data set (this corresponds to two slots crossing the origin).

Since the data form two (connected) branches, more than one initialization is needed. We choose to initialize four starting points. We apply an angle penalization using $k = 2$, which serves to keep the curve on the correct slot at the crossing. The result is shown in Fig. 6.

5 Theoretical background

In this section we compare our algorithm to that of Delicado in more depth. The analogy between the two algorithms becomes clearer when the directions used in the two algo-

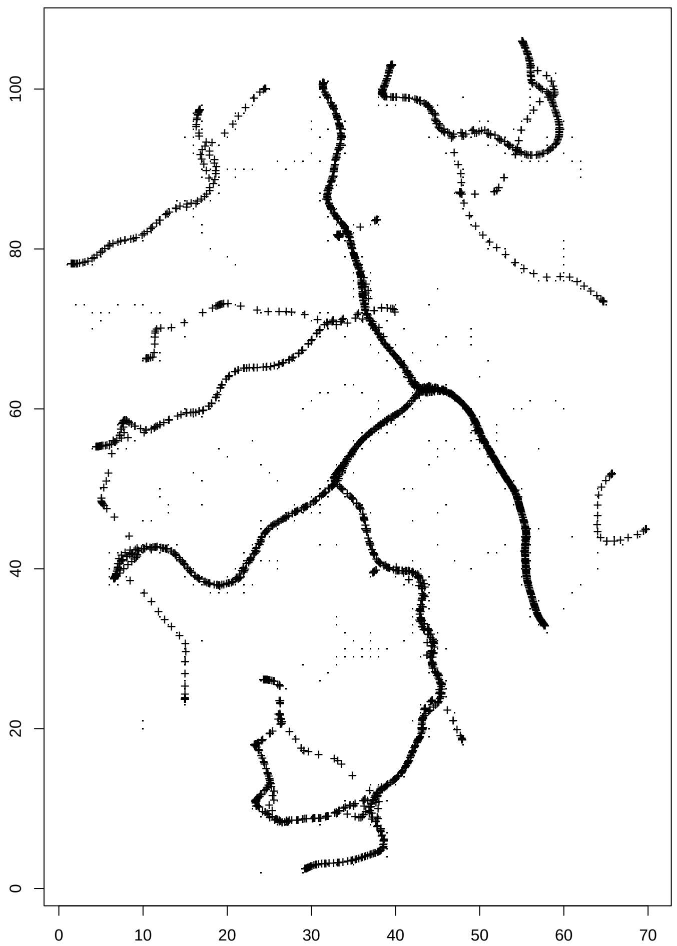

Figure 5: Floodplain data (.) with principal curves (+).

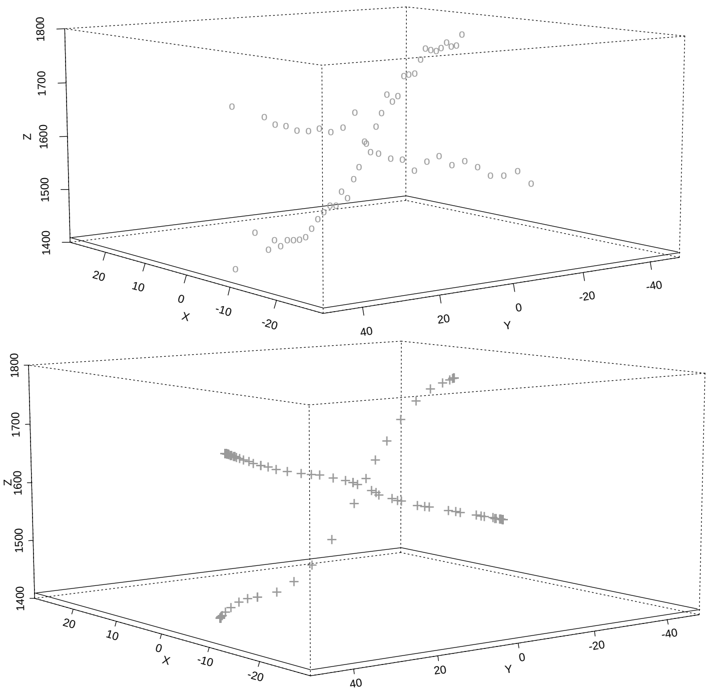

Figure 6: Galaxy data (o) with principal curves (+).

rithms are compared. Both can be shown to fulfil rather similar optimality properties. The first local principal component γ^x as used in our algorithm is the eigenvector of Σ^x corresponding to the largest eigenvalue, i.e. the γ maximizing $\gamma^T\Sigma^x\gamma$. Let B_γ be an orthonormal basis of the orthogonal complement of the subspace spanned by γ. As we have for all $\gamma \in \mathbb{R}^d$ that $\mathrm{tr}(\Sigma^x) = \gamma^T\Sigma^x\gamma + \mathrm{tr}\left(B_\gamma^T\Sigma^x B_\gamma\right)$, we can equivalently minimize

$$\mathrm{tr}\left(B_\gamma^T\Sigma^x B_\gamma\right). \tag{7}$$

Recalling the denotation from the introduction, we have

$$\tilde{\mu}(x,b) = \sum_{i=1}^n \tilde{w}_i X_i^H, \qquad \text{with} \qquad \tilde{w}_i := \frac{\bar{w}_i c_i}{\sum_{j=1}^n \bar{w}_j c_j},$$

where c_i is an indicator taking the value 1 if X_i^H is in the cluster of data around x. Delicado defines his principal direction $\tilde{b}^*(x)$ to be orthogonal to the $(d-1)$-dimensional subspace within which the (conditional) total variance is minimal, i.e. he seeks the b that minimizes

$$
\begin{aligned}
&\widehat{TV}(X|X \in H(x,b)) = \\
&= \sum_{i=1}^{n} \tilde{w}_i \cdot (X_i^H)^T X_i^H - \tilde{\mu}(x,b)^T \tilde{\mu}(x,b) = \\
&= \operatorname{tr}\left(\sum_{i=1}^{n} \tilde{w}_i \left(X_i^H - \tilde{\mu}(x,b) \right) \left(X_i^H - \tilde{\mu}(x,b) \right)^T \right) = \\
&= \operatorname{tr}\left(\sum_{i=1}^{n} \tilde{w}_i \left(x + B_b B_b^T (X_i - x) - \tilde{\mu}(x,b) \right) \left(x + B_b B_b^T (X_i - x) - \tilde{\mu}(x,b) \right)^T \right) = \\
&= \operatorname{tr}\left(B_b B_b^T \left(\sum_{i=1}^{n} \tilde{w}_i (X_i - \tilde{\mu}(x,b))(X_i - \tilde{\mu}(x,b))^T \right) B_b B_b^T \right) = \\
&= \operatorname{tr}\left(B_b^T \tilde{\Sigma}^x(b) B_b \right), \qquad (8)
\end{aligned}
$$

with

$$\tilde{\Sigma}^x(b) = \left(\sum_{i=1}^{n} \tilde{w}_i (X_i - \tilde{\mu}(x,b))(X_i - \tilde{\mu}(x,b))^T \right).$$

The semiorthogonal matrix B_b is defined in analogy to B_γ. When comparing expression (7), which defines the local principal component, to expression (8) as used by Delicado, one can observe that in both cases some sort of ("pseudo") covariance matrix is involved. The two matrices Σ^x and $\tilde{\Sigma}^x(b)$ differ in the type of centering and the type of weighting used. But Delicado's weights depend on b, as Delicado does not directly use the distance between the X_i and x but only the part hereof which is parallel to b. Therefore the "pseudo" covariance matrix $\tilde{\Sigma}^x(b)$ depends on b and the principal direction of Delicado cannot be obtained using an eigen decomposition. This makes Delicado's algorithm — especially compared to the local principal curve algorithm, which is based on an eigen decomposition — numerically quite complex. If a small cluster size is used in Delicado's algorithm, then the difference between the weights w_i and $\tilde{w}_i$ gets small enough so that the local principal curve algorithm can be seen as a simple and fast approximation to Delicado's nice, but complex algorithm. Compare for illustration Fig. 1 in this paper with Fig. 3 in Delicado & Huerta (2003).

6 Coverage

There is a need for some criterion that evaluates the performance of a principal curve. This is usually done by means of a quantitative measure as the expected squared distance

$$\triangle(m) = E\left(\inf_t ||X - m(t)||^2\right) \tag{9}$$

between a random variable X and the curve m. Principal curves according to HS are critical points of (9) (Duchamp & Stuetzle (1996) even show that they are always saddle-points), whereas principal curves in the sense of KKLZ minimize (9) over a class of curves with bounded length. Another quantitative measure, which is closely related to $\triangle(m)$, is the proportion of the generalized total variance not explained by the principal curve (Delicado, 2001). Alternatively, one can consider the *coverage* of a principal curve m, being defined by the fraction of all data points which are found in a certain neighborhood of the principal curve. More precisely, let an algorithm select a principal curve m consisting of a set of points P_m. Then

$$C_m(\tau) = \#\{x \in \{X_1, \ldots, X_n\} | \exists p \in P_m \text{ with } ||x - p|| \leq \tau\}/n$$

is the coverage of curve m with parameter τ. Obviously the coverage is a monotonically increasing function of τ and will reach the value 1 for τ tending to infinity. Note that the coverage can be interpreted as the empirical distribution function of the "residuals", i.e. the shortest distance between data and principal curve. For evaluating the performance of a principal curve fit it is necessary to take a look at the whole coverage curve $C_m(\tau)$. In Fig. 7 we provide the coverage plots for the examples given in Fig. 2 and 3. For data with moderate noise, which are depicted in the left column, the results from KKLZ and LPC are comparable, except for the big spiral, where LPC performs obviously better. In the right column, where the noisy data are examined, LPC is always among the best, but is slightly beaten twice by KKLZ (small spiral, zigzag).

Certainly a concave coverage curve is desirable, i.e. it is "best" when rising rapidly for small τ. The better the principal curve, the smaller is the area in the left top above the coverage curve, i.e. the area between $C_m(\tau)$, the line $\tau = 0$ and the constant function $c(\tau) = 1$. This area, which corresponds to the mean length of the observed residuals, can be seen as an estimator of

$$E\left(\inf_t ||X - m(t)||\right),$$

which is the L_1 version of $\triangle(m)$. To obtain a quantitative measure for the performance of a principal curve, we set this area in relation to the corresponding area obtained by standard

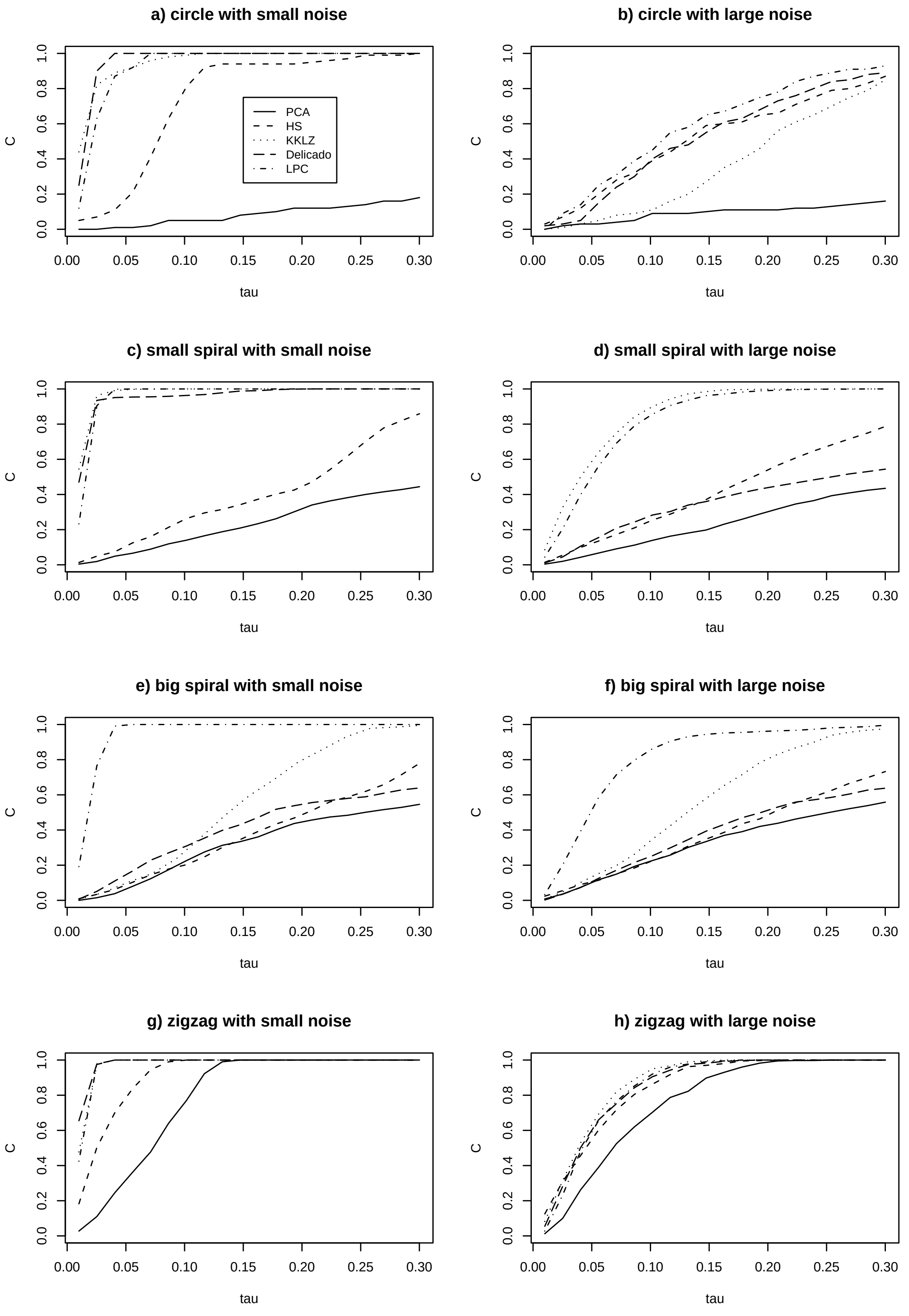

Figure 7: Coverages for simulated data from Fig. 2 and 3.

principal component analysis. The smaller this quotient A_C, the smaller is the relative mean length of the observed residuals and the better is the principal curve compared to principal components. The following table provides the value $R_C = 1 - A_C$ for HS, KKLZ, Delicado, and LPC, for all simulated examples treated above:

A_C	$a)$	$b)$	$c)$	$d)$	$e)$	$f)$	$g)$	$h)$
HS	0.71	0.46	0.28	0.23	0.08	0.08	0.59	0.34
KKLZ	0.95	0.29	0.97	0.80	0.50	0.35	0.88	0.45
Delicado	0.95	0.47	0.95	0.15	0.13	0.08	0.92	0.38
LPC	0.92	0.54	0.95	0.76	0.92	0.71	0.87	0.37

Table 1: R_C for scenarios a) to h) and various principal curve algorithms.

For the HS algorithm, the value R_C frequently takes values near 0, which means a quite bad performance. KKLZ wins in four of our examples, Delicado has the best performance in two situations, while LPC is the best algorithm for three scenarios. Note that there are two cases (e and f) where LPC is the only algorithm yielding an useful result, and that in cases, when it is beaten by another algorithm, it performs only slightly worse than the winner.

It should be remarked that the measure R_C can be interpreted in the same spirit as the coefficient of determination R^2 used in regression analysis: Values near 1 indicate a good fit, while values near 0 mean bad performance. Like R^2, this criterion is certainly only a suitable tool to compare (principal) curves which are sufficiently smooth, since a principal curve interpolating the data has constant coverage 1, and thus $R_C = 1$. In order to obtain an objective criterion to compare the performance of principal curves of different smoothness, some kind of penalization for R_C has to be introduced. We will not follow this direction further, but focus on a modified version of the coverage, where the fixed curve m is replaced by principal curves $m(\tau)$ calculated by employing the coverage parameter τ as bandwidth. In this manner one obtains the *self-coverage*

$$S(\tau) = C_{m(\tau)}(\tau),$$

which compares different bandwidths for one specific algorithm rather than being a tool for the comparison of different principal curve algorithms. For scenarios a) to h), we calculate $S(\tau)$ over a grid from $\tau = 0.01$ up to $\tau = 1.0$ in steps of 0.01, where $m(\tau)$ is obtained via the LPC algorithm. The results are presented in Fig. 8. One observes that the self-coverage often has a typical behavior: It starts with small values, then increases rapidly until a local maximum is reached, where the best fit is achieved. Afterwards the self-coverage curve falls again, possibly showing erratic behavior, or remains on a constant level, but finally always reaches the value 1.

Intuitively, if a certain bandwidth τ is suitable for the calculation of m, then the same τ should adequately cover the width of the data cloud around the principal curve $m(\tau)$, i.e. lead to a high self-coverage at this value τ. The bandwidth h then may be selected as the value where $S(\tau)$ achieves its first distinct local maximum, or, if no local maximum exists, as the first value where the function $S(\tau)$ achieves the value 1.

Fig. 8 demonstrates how this parameter selection rule has to be interpreted: In cases c) to g) the situation is obvious. If the first maximum is a plateau as in c) or e), one simply has to choose the first value of this plateau. In situation a) obviously not the little peak at the very beginning is the appropriate one, but $h = 0.17$. Most difficult are data situations where the underlying structure is quite simple, but contaminated with large noise as in b) or h). Then one usually does not get a clear distinguishable first maximum, but rather a sequence of little plateaus, and one has to try with various bandwidths in these cases. Generally speaking, the performance of this parameter selection rule increases with decreasing noise and increasing complexity of the underlying structure.

In all these computations, we worked with one fixed starting point, since the data clouds are connected and consist of only one branch. Applying this bandwidth selection method to the floodplain data (with multiple initializations) yields $h = 2.5$. The resulting principal curve (not depicted here) seems suitable, too, but *knowing that the underlying curves should be rivers*, we decided for the smaller value $h = 1.5$ in this case. In practice, a notion about the desired shape of the resulting fit should always permit to take a critical look on the algorithmically selected bandwidths.

7 Discussion

We demonstrated that the concept of applying local principal components in connection with the mean shift is a simple and useful tool for computing principal curves, which shows superior performance in simulated data sets compared to most of the other principal curve algorithms. We showed that the algorithm works in simulated and real data sets even for highly complicated data structures. This includes data situations which by existing methodology could only be handled unsatisfactorily, as data with multiple or disconnected

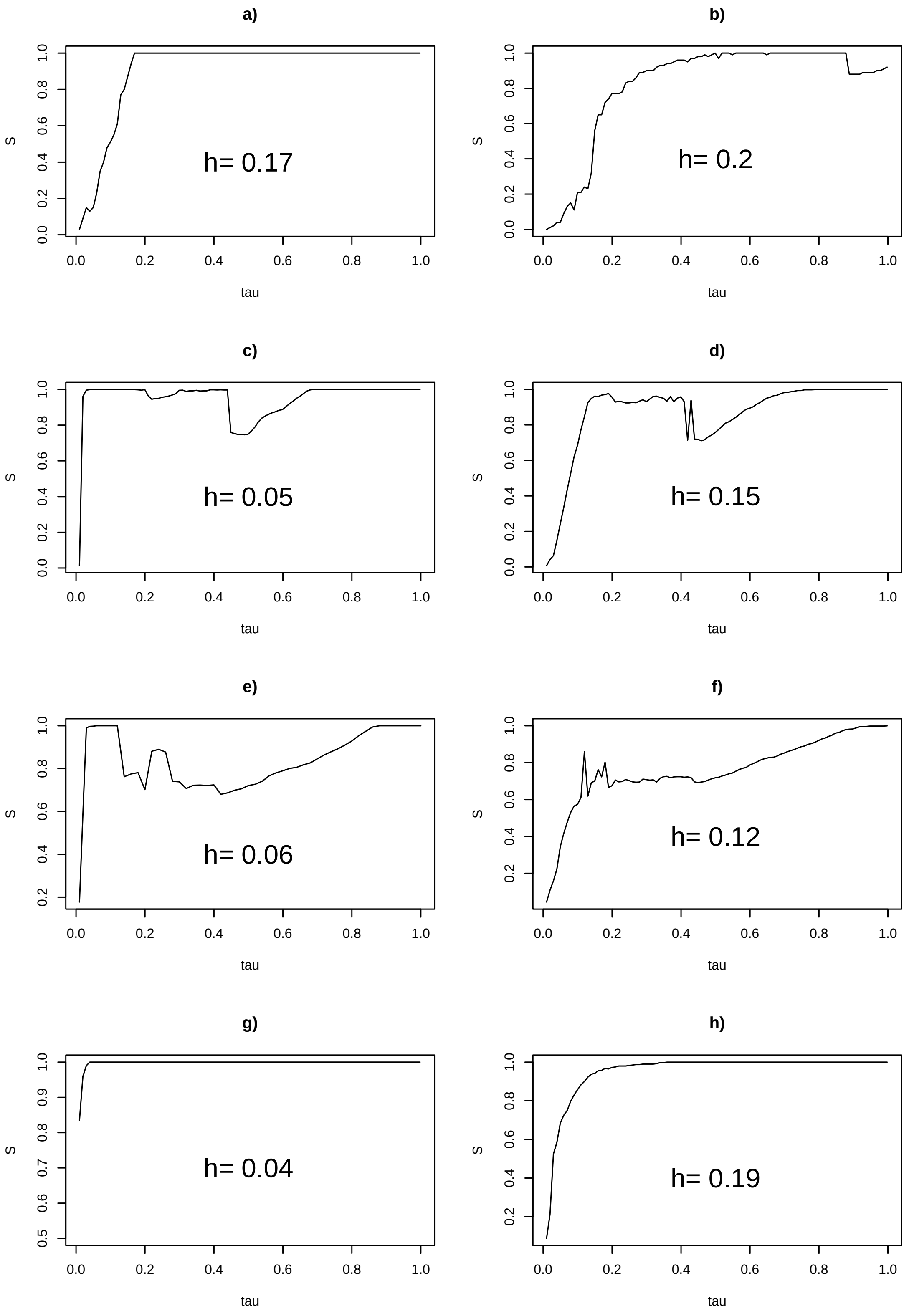

Figure 8: Self-coverages for situations a) to h) and selected bandwidths.

branches. Especially for noisy spatial data like the floodplain data the approach has a high potential to detect the underlying structure. We further provided an approach to select the necessary parameters in a data-adaptive way, but it has to be pointed out that bandwidth selection still requires further attention, particularly for noisy data structures.

There is still need for further research concerning the theoretical background of the method. Although working fine, we do not have much of a theoretical justification why we move along the data cloud with *local principal components*. This choice is sensible but in no way unique, and there seem to be many alternatives, such as the extrapolation of the already estimated part of the curve. Due to the nice properties of the mean shift, it might even work to use a line in an *arbitrary* direction, as long it is not orthogonal to the principal curve in the observed point. It is crucial that a movement is made - the mean shift will afterwards adjust the principal curve in direction of the "middle" of the data cloud. However, by applying local principal components the algorithm is fastest, most stable, and the results are as intuitively expected. We consider the first local principal component to be a (biased) approximation of the tangent to the crest line of the estimated density. One can easily derive from its definition that the first local principal component around x is the line through μ^x which minimizes the weighted distance between the X_i and the line, using the weights w_i as in (4). The first local principal component is therefore the line that locally gives the best fit.

Furthermore, it will be interesting to investigate if the proposed algorithm can be extended to obtain local principal surfaces or even local principal manifolds of higher dimensions. This might be quite difficult, since easy techniques as the signum flipping or the mean shift are probably not transferable to higher dimensional curves without developing new concepts. The problem of assigning a low-dimensional coordinate system to a high-dimensional sample space ("charting") is currently discussed intensively in machine learning, see e.g. Brand (2003).

Acknowledgements

We gratefully acknowledge support from Deutsche Forschungsgemeinschaft (Sonderforschungsbereich 386: Statistical Analysis of Discrete Structures).

References

Aluja-Banet, T. and Nonell-Torrent, R. (1991). Local principal component analysis. *Qüestioó* **3**, 267–278.

Banfield, J. D. and Raftery, A. E. (1992). Ice flow identification in satellite images using mathematical morphology and clustering about principal curves. *J. Amer. Statist. Assoc.* **87**, 7–16.

Brand, M. (2003). Charting a manifold. *Proceedings Neural Information Processing Systems* **15**, .

Buta, R. (1987). The structure and dynamics of ringed galaxies, III: Surface photometry and kinematics of the ringed nonbarred spiral NGC 7531. *The Astrophysical Journal Supplement Series* **64**, 1–137.

Chang, K. and Ghosh, J. (1998). Principal curves for nonlinear feature extraction and classification. *SPIE Applications of Artificial Neural Networks in Image Processing III* **3307**, 120–129.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Machine Intell.* **24**, 603–619.

Delicado, P. (2001). Another look at principal curves and surfaces. *Journal of Multivariate Analysis* **77**, 84–116.

Delicado, P. and Huerta, M. (2003). Principal curves of oriented points: Theoretical and computational improvements. *Computational Statistics* **18**, 293–315.

Duchamp, T. and Stuetzle, W. (1996). Extremal properties of principal curves in the plane. *Ann. Statist.* **24**, 1511–1520.

Hastie, T. and Stuetzle, W. (1989). Principal curves. *J. Amer. Statist. Assoc.* **84**, 502–516.

Jolliffe, I. T. (2002). *Principal Component Analysis, Second Edition.* New York: Springer.

Kambhatla, N. and Leen, T. K. (1997). Dimension reduction by local PCA. *Neural Computation* **9**, 1493–1516.

Kégl, B. and Krzyzak, A. (2002). Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**, 59–74.

Kégl, B., Krzyzak, A., Linder, T., and Zeger, K. (2000). Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 281–297.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space.

Philosophical Magazine **2**, 559–572.

Schölkopf, B. and Smola, A. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10**, 1299–1319.

Skarbek, W. (1996). Local principal components analysis for transform coding. In *Int. Symposium on Nonlinear Theory and its Applications - NOLTA'96, Proceedings*, Katsurahara-so, Japan, pp. 413–416.

Tharpey, T. and Flury, B. (1996). Self-consistency: A fundamental concept in statistics. *Statistical Science* **11**, 229–243.

Tibshirani, R. (1992). Principal curves revisited. *Statistics and Computing* **2**, 183–190.

Verbeek, Vlassis, N., and Kröse, B. (2001). A soft k-segments algorithm for principal curves. In *Proceedings International Conference on Artificial Neural Networks*, Vienna, Austria, pp. 450–456.

Wand, M. P. and Jones, M. C. (1993). Comparison of smoothing parametrizations in bivariate kernel density estimation. *J. Amer. Statist. Assoc.* **88**, 520–528.