

# Principal curves and their use in FDA

Stanislav Kolenikov  
skolenik@unc.edu

117 New West, Cameron Ave, University of North Carolina,  
Chapel Hill, NC 27599-3260, US

March 23, 2001

## 1 Theory... or, rather, the idea

The idea behind the principal curves is the one of the curve passing through the bulk of the data. They are different from say principal components in that they allow for nonlinearities in the data cloud, although principal components are the principal curves for elliptical (including of course multivariate Gaussian) distributions. They are also different from regression smoothers — or, rather, the relation between principal curves and nonparametric regression is roughly the same as between linear regression and principal components. The regression approaches distinguish one variable as the dependent one, while the principal components / curves do not distinguish between the variables they rely on. This is very well applicable in the context of the functional data that come measured in the same units and of comparable magnitude in all dimensions.

The basic reference on the topic is [Hastie and Stuetzle (1989)] (HS further on), and it actually starts with making the distinction between the four aforementioned concepts. HS define *principal curves* as the curves self-consistent with respect to projection of the population or data points. Two concepts are effectively used in this definition: projection and conditional expectation.

If the curve is parametrized with the parameter  $\lambda$ , then the *projection index* of a data point is the arg of the point on the curve to which the data point is projected:

$$\lambda_{\mathbf{f}}(\mathbf{x}) = \sup_{\lambda} \left\{ \lambda : \|\mathbf{x} - \mathbf{f}(\lambda)\| = \inf_{\mu} \|\mathbf{x} - \mathbf{f}(\mu)\| \right\}, \quad (1)$$

i.e. the value of  $\lambda$  for which  $\mathbf{f}(\lambda)$  is closest to  $\mathbf{x}$ .

With this projection index, self-consistent / principal curves are the curves such that

$$E(\mathbf{X} | \lambda_{\mathbf{f}}(\mathbf{X}) = \lambda) = \mathbf{f}(\lambda) \quad (2)$$

HS note that with this definition, if the data are actually generated by  $\mathbf{X} = \mathbf{f}(\lambda) + \varepsilon$ , then  $\mathbf{f}$  need not be the principal curve. Imagine a circle with the points distributed normally around its circumference. Then at each point, there is more mass outside the circle, so the principal curve would go outside the data generating circle. The situation is generic for  $\mathbf{f}(\cdot)$  that has curvature.

An alternative definition is proposed by [Tibshirani (1992)]. His data generating mechanism has two stages: 1. generating a latent variable  $S$  according to some distribution  $g_S(s)$ , and then 2. generating  $\mathbf{Y} \in \mathbf{R}^p$  from the conditional distribution  $g_{\mathbf{Y}|s}$ . Then Tibshirani's definition of the principal curves is a triple  $\langle g_S, g_{\mathbf{Y}|s}, \mathbf{f} \rangle$  satisfying

I.  $g_{\mathbf{Y}}(y) = \int g_{\mathbf{Y}|s} g_S(s) ds;$

- II.  $Y_1, \dots, Y_p$  conditionally independent given  $s$ ;
- III.  $\mathbf{f} : \Gamma \rightarrow \mathbf{R}^p$ ,  $\Gamma$  is a closed interval in  $\mathbf{R}$ , and  $E\mathbf{Y}|s = \mathbf{f}(s)$ .

This definition does not suffer from bias, but it only coincides with HS if the support of the conditional distribution  $g_{\mathbf{Y}|s}$  is orthogonal to the curve  $\mathbf{f}(\cdot)$  at  $s$ .

The HS definition naturally gives rise to the following algorithm for finding the principal curves.

1. Start from the first principal component:

$$\mathbf{f}^{(0)} = \bar{\mathbf{x}} + \mathbf{a}\lambda, \lambda^{(0)}(\mathbf{x}) = \lambda_{\mathbf{f}^{(0)}}(\mathbf{x}). \quad (3)$$

2. Set

$$\mathbf{f}^{(j)}(\cdot) = E(\mathbf{X} | \lambda_{\mathbf{f}^{(j-1)}}(\mathbf{X}) = \cdot) \quad (4)$$

Done by scatterplot smoother: perform local fitting for each dimension.

3. Define

$$\lambda^{(j)}(\mathbf{x}) = \lambda_{\mathbf{f}^{(j)}}(\mathbf{x}) \quad (5)$$

and transform to unit speed parameterization.

4. Evaluate

$$\Delta^{(j)} = E \left[ \|\mathbf{X} - \mathbf{f}(\lambda^{(j)}(\mathbf{X}))\|^2 \right] \quad (6)$$

5. Stop if  $\Delta^{(j)}$  is small enough, otherwise  $j \leftarrow j + 1$  and go to step 2.

The algorithm that [Tibshirani (1992)] comes up with is a version of the EM-algorithm, and the final formulas coincide with those for a final mixture of normal distributions. In fact, this mixture has a support on at most  $n$  points which are essentially the projection points of the data. The principal curves produced by this algorithm are a bit shorter than those produced by the HS version, in the sense that they do not protract as far as HS curves into the "ends" of distributions in the examples provided.

## 2 My implementation

The HS algorithm was implemented in Stata (version 7). The code is available from me, and it will be available from my website upon finalization.

The implementation relies on the lowess smoother built-in into Stata at the step 2 of the HS algorithm. The bandwidth can be varied somewhat as one of the program options. Other options include trivial things like convergence criteria, maximum number of iterations, some visualisation options, and some robustness options (see below in the discussion of examples).

I discussed some of the issues with Trevor Hastie over e-mail, and I include his comments along with my text where applicable.

## 3 Examples

The HS paper provides an example of the data generated as points distributed uniformly on a circle with Gaussian noise added. For this data, the algorithm should be able to find this circle, or something about it, given the bias considerations. In fact, any diameter of the circle is the principal curve for the population, too.

Figure 1 provides an example of the iterations with those circular data<sup>1</sup>. Also, the log of iterations might be of interest.

---

<sup>1</sup> The figures are a bit off scale: the vertical and the horizontal axes are not in the same units.

Iteration 1:	sum of squares =	32.446591;	length =	15.01039
Iteration 2:	sum of squares =	14.693489;	length =	15.411715
Iteration 3:	sum of squares =	10.048519;	length =	20.143418
Iteration 4:	sum of squares =	7.1063338;	length =	15.178684
Iteration 5:	sum of squares =	6.9243142;	length =	16.930701
Iteration 6:	sum of squares =	5.6747848;	length =	10.976699
Iteration 7:	sum of squares =	6.1515575;	length =	16.491178
Iteration 8:	sum of squares =	5.584682 ;	length =	18.721084
Iteration 9:	sum of squares =	6.3345229;	length =	14.008519
Iteration 10:	sum of squares =	6.5243196;	length =	18.205329
Iteration 11:	sum of squares =	7.0125932;	length =	18.065242
Iteration 12:	sum of squares =	6.8085226;	length =	15.256001
Iteration 13:	sum of squares =	7.4998452;	length =	18.063828
Iteration 14:	sum of squares =	7.174593 ;	length =	17.184668
Iteration 15:	sum of squares =	7.2292064;	length =	22.154814
Iteration 16:	sum of squares =	9.3359484;	length =	16.987542
Iteration 17:	sum of squares =	7.7802278;	length =	21.893586
Iteration 18:	sum of squares =	10.688608;	length =	29.092229
Iteration 19:	sum of squares =	6.3593014;	length =	20.434345
Iteration 20:	sum of squares =	4.8465258;	length =	10.615708
Iteration 21:	sum of squares =	4.5149404;	length =	10.175274
Iteration 22:	sum of squares =	4.4132195;	length =	7.7543573
Iteration 23:	sum of squares =	4.4237536;	length =	5.7521201
Iteration 24:	sum of squares =	4.3374652;	length =	6.075431
Iteration 25:	sum of squares =	4.2372211;	length =	6.9130063
Iteration 26:	sum of squares =	4.380806 ;	length =	7.1375016
Iteration 27:	sum of squares =	4.3937391;	length =	6.4160796
Iteration 28:	sum of squares =	4.41773 ;	length =	6.5763004
Iteration 29:	sum of squares =	4.3218421;	length =	6.4826352
Iteration 30:	sum of squares =	4.3509631;	length =	6.4628284

I stopped the program when visual convergence was achieved. The SS reported on the graph seem to be misaligned by one iteration; that of course can be fixed.

*T. Hastie: The S in the circle does sound familiar. That is a tough problem, and starting with a line is rather brutal. Nevertheless, I don't recall having too hard a time with it.*

The second example I made up was that of parabolas, where the second component is a quadratic function of the first one, with heteroskedastic Gaussian errors added. An example of raw data are presented at Fig. 2; the seed indicates the random seed of Stata random number generator. (For each new value of the seed, a new sample was generated; probably I should have stuck to a single data set instead.) The iterations are given at Fig. 3. Apparently, this data set came to be difficult for the (naive implementation of the) algorithm: the algorithm is captured by a strange curve that is not changing with iterations.

*T. Hastie: I certainly NEVER encountered the problems you have with the parabola.*

A bit different pictures can be obtained in other simulation samples (i.e. with different random number generator seeds), though, where the curve that I would like to see was obtained (Fig. 4). In fact, it grew out of what appeared to be an artefact on Fig. 3.

I found three solutions to the problem. The first is a purely algorithmic one. I introduced a check for zero density along the current version of the curve, and if points with zero density were found, then the curve was trimmed so that the line connecting the data along the diameter of the data points set (and essentially missing any data points) is eliminated, and parametrization along one of the remaining branches is reversed.

Another remedy, also of algorithmic nature, was to specify a starting curve other than the principal component. Two obvious choices were implemented: starting from random connection between the data points, and starting from a line parametrized by one of the variables. With the parabola example, the

natural parametrization is with  $x_1$ . The results are reported in Fig. 5 and Fig. 6 for random start and for  $x_1$  start, respectively.

Finally, an elegant and simple statistical solution was found to increase the bandwidth of the lowess smoother — see Fig. 7.

Of course, the results with the parabola data cast some doubt on the performance of the (my implementation of) HS principal curves algorithm. It looks as if some additional effort and art might be required to achieve convergence to nice results. Also, the sum of squares does not decrease monotonically (as it probably would with Tibshirani (1992) EM type version of the algorithm). I attribute some of the variability in the sum of squares to the swings of the ends of the curve.

*T.Hastie: While monotone decreasing SS is not guaranteed, I don't recall it being much of a problem. I typically used a fixed bandwidth. However, a reasonable strategy would be to start off with a big bandwidth, let it converge, then reduce, let it converge and so on.*

## 4 Application to functional data

The final goal of the development of this piece of software was to apply it to the functional data ([Ramsey and Silverman (1997)]) problem. The early application is based on the set of simulated data. There are 84 curves measured at 41 points (a grid of 0.05 on  $[-2, 2]$ ). How many features of the functional data can you distinguish at Fig. 8? :) Well, the details of simulation will be hidden until all results are presented...

The principal components analysis produced rather sensible results. The first four components are shown at Fig. 9. The first PC explains 73.2% of the residual variation (i.e. after subtracting the mean); the second PC, 26.5%; the third PC, 0.04%; and the fourth PC, 0.02%. It appears from the graph that the first two components do capture some large scale differences between the curves, the third one is a vertical shift, and the fourth one is a pure noise (as can be seen with some magnification, believe me :). It is a bit troublesome, however, that the mean + first component curve goes beyond the actual data points, the envelope of which is shown with circles.

Now, what was the performance of the principal curve method? As I've seen before, some track of performance is quite desirable, but the question is, how do I plot 41-dimensional data? I chose several bivariate plots to be having a look at how things go. The first few iterations are shown at Fig. 10–13. The relative failure of the principal components is evident on the first of those graphs, and the success of the principal curves algorithm, on the rest of them :).

The iteration log follows:

```
Iteration 0: length =          10.729539
Iteration 1: sum of squares =   28.637694

Iteration 1: length =          43.212788
Iteration 2: sum of squares =   16.514586

Iteration 2: length =          19.477486
Iteration 3: sum of squares =   11.628214
```

Fig. 14 show the projections found by the principal curve method in the data space. The curves shown roughly correspond to the 5%, 25%, 50%, and 90% of the  $\lambda$  (projection index) sample distribution. As compared to the principal components, the principal curve captured large scale variations in the shapes, but not the additive shift.

To what extent does this picture agree with the data generating model? The truth is that the data were generated according to the following model:

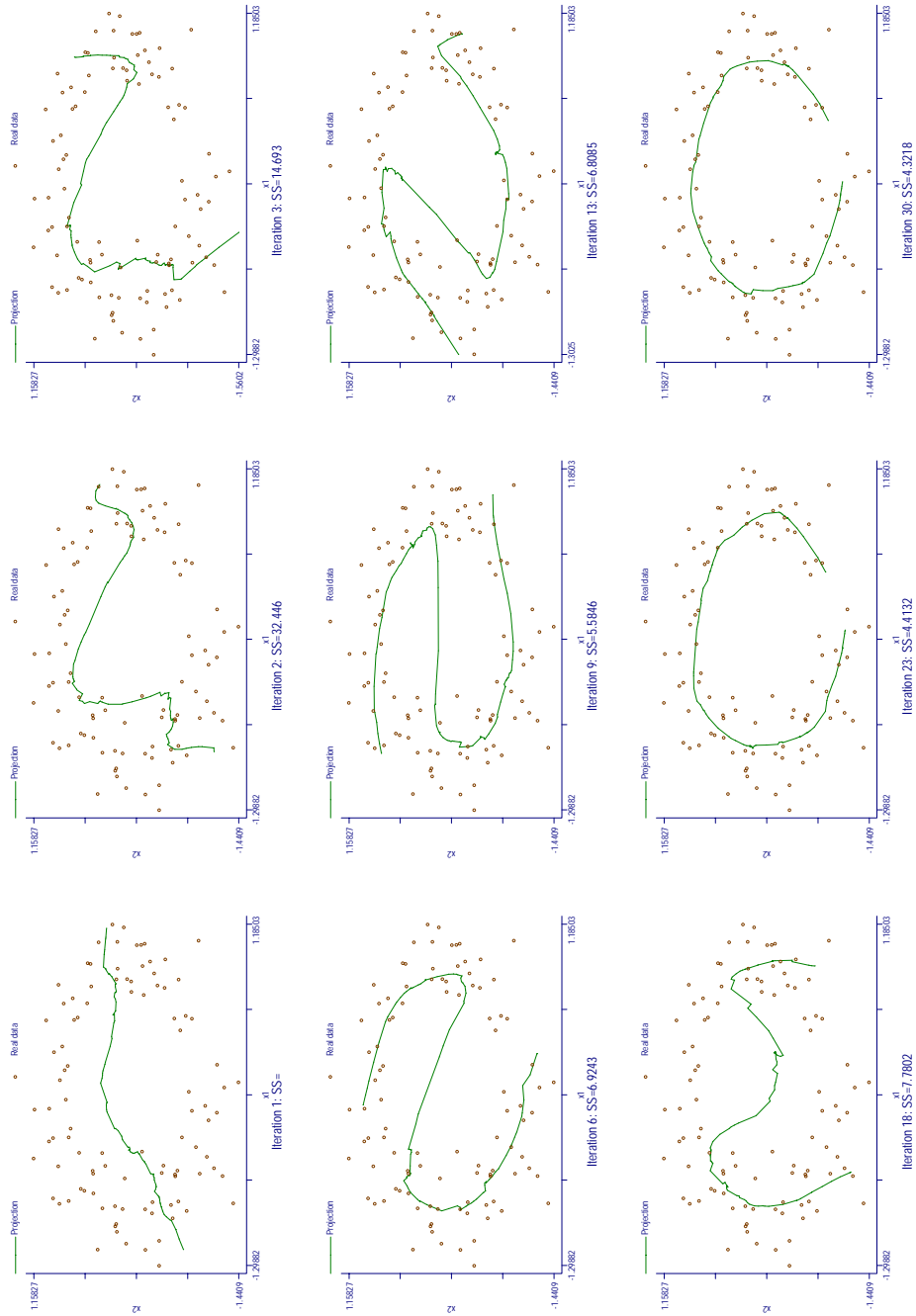
$$x_{ik} = 1 + t - 2\beta_k \exp(-3 * \alpha_k t^2) - 2\gamma_k I_{t>0} + \delta_k + \varepsilon_{ik}, \quad t = (i - 21)/20, k = 1, \dots, 84, \quad (7)$$

where  $i$  and  $t$  correspond to the measurement points / dimensions, and  $k$  enumerates observations / curves,  $\delta_k \sim N(0, 0.03^2)$ ,  $\varepsilon_{ik} \sim N(0, 0.02^2)$  are errors independent of each other and of anything else. The parameters  $\alpha_k, \beta_k, \gamma_k$  are distributed uniformly inside a tube that goes along the edges of the unit cube — see Fig. 15. So, roughly 30 points were generated for  $\alpha_k \sim U[0, 1]$ , with low  $\beta_k$  and  $\gamma_k$  i.i.d  $U[0, 0.1]$  — thus, those curve would exhibit varying width of the valley, but its magnitude would be seriously dampened by the  $\beta_k$  coefficient. Another 30 or so points were generated along the edge  $\beta_k \sim U[0, 1]$  with high  $\alpha_k \sim U[0.9, 1]$  and low  $\gamma_k \sim U[0, 0.1]$  — i.e., pronounced width of the valley, varying depth, and small kink. Finally, the last 30 or so points were generated along the  $\gamma_k \sim U[0, 1]$  axis with both  $\alpha_k$  and  $\beta_k$  high (i.i.d.  $U[0.9, 1]$ ): the valley was wide and deep, and the kink expressed itself to a varying degree.

This is essentially my feature space (except for the  $\delta$ s), and I was wondering whether the principal curve algorithm would find it. To my assessment, it did! It is also worth noting that the second principal component captured two of the features — the hump related to  $\alpha_k$  and  $\beta_k$ , and the kink related to  $\gamma_k$ , so it essentially went along the  $\gamma$  axis in the 3D feature space.

## References

- [Dempster et. al. (1977)] Dempster, A.P., Laird N.M., and Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *JRSS B*, **39**, 1–38 (1977).
- [Hastie and Stuetzle (1989)] Hastie, T., and Stuetzle, W. Principal Curves. *JASA*, **84**, 502–516 (1989).
- [Tibshirani (1992)] Tibshirani, R. Principal Curves Revisited.  
<http://www-stat.stanford.edu/~tibs/ftp/princcurve.ps>
- [Ramsey and Silverman (1997)] Ramsey, J.O., and Silverman, B.W. Functional Data Analysis (Springer Series in Statistics). Springer (1997).



Circular data; seed = 3940

Figure 1: Circular data example

Figure 2: Raw parabola data; the first PC overlaid.

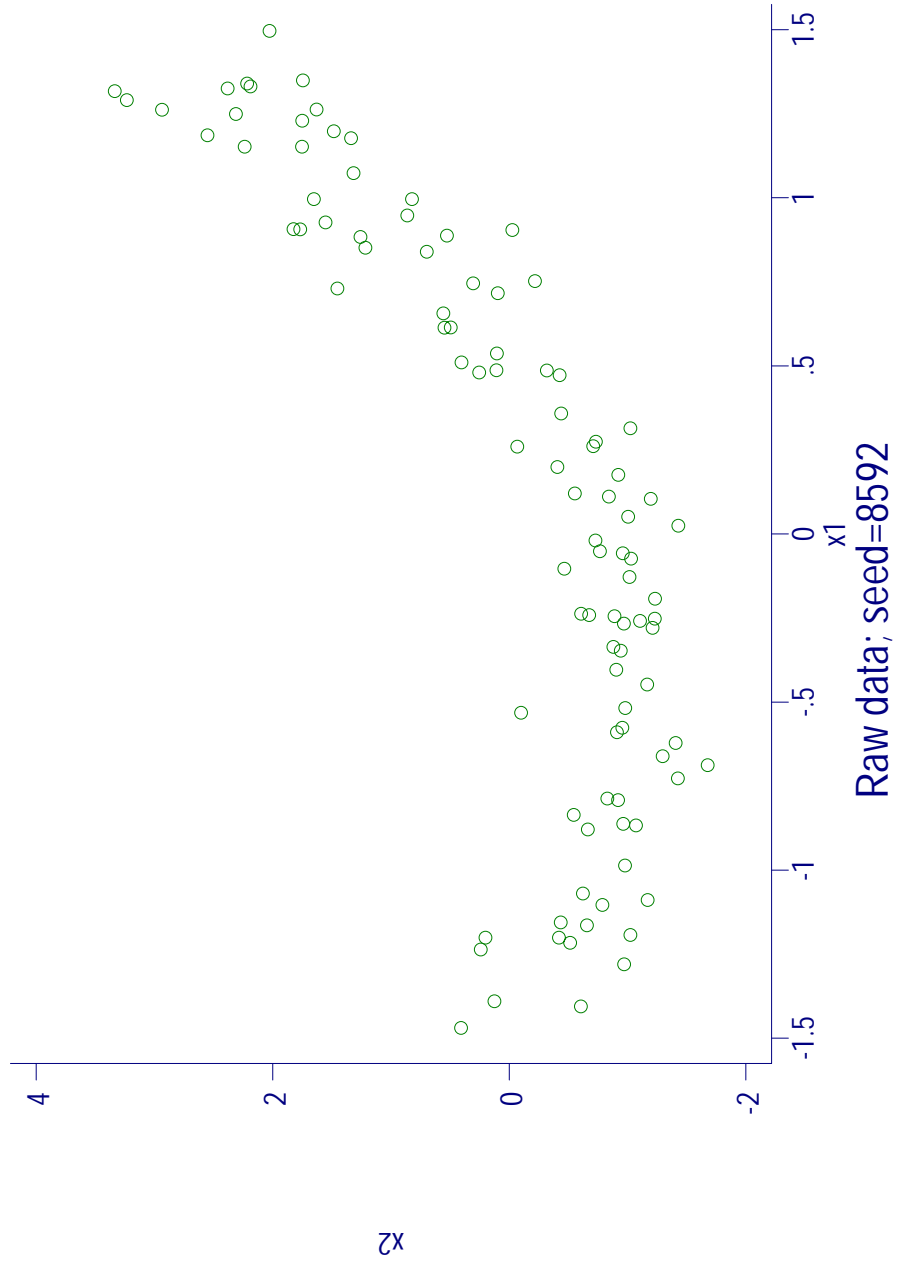
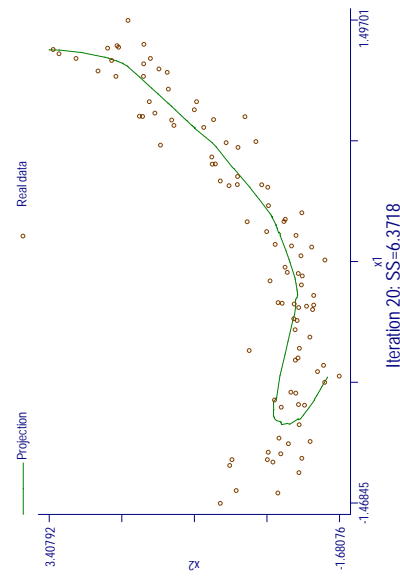
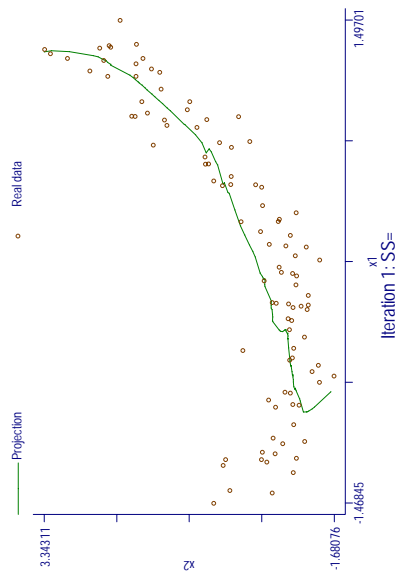
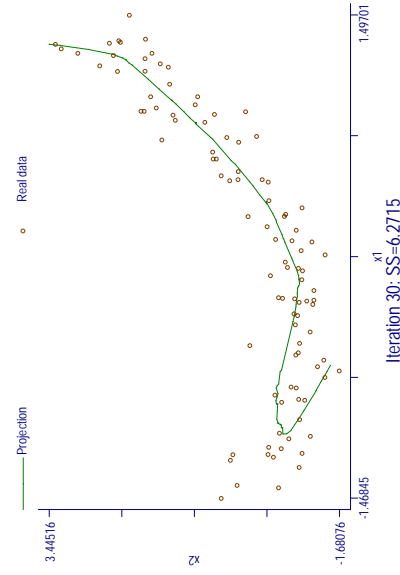
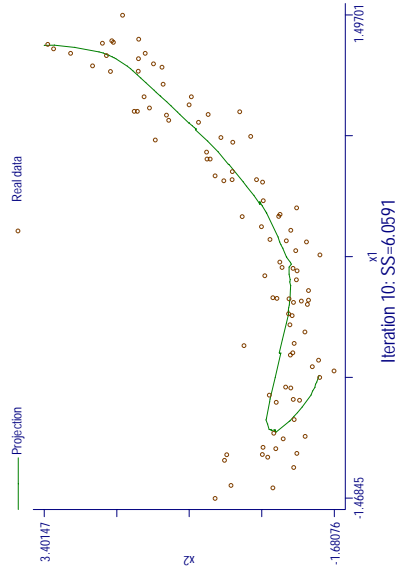
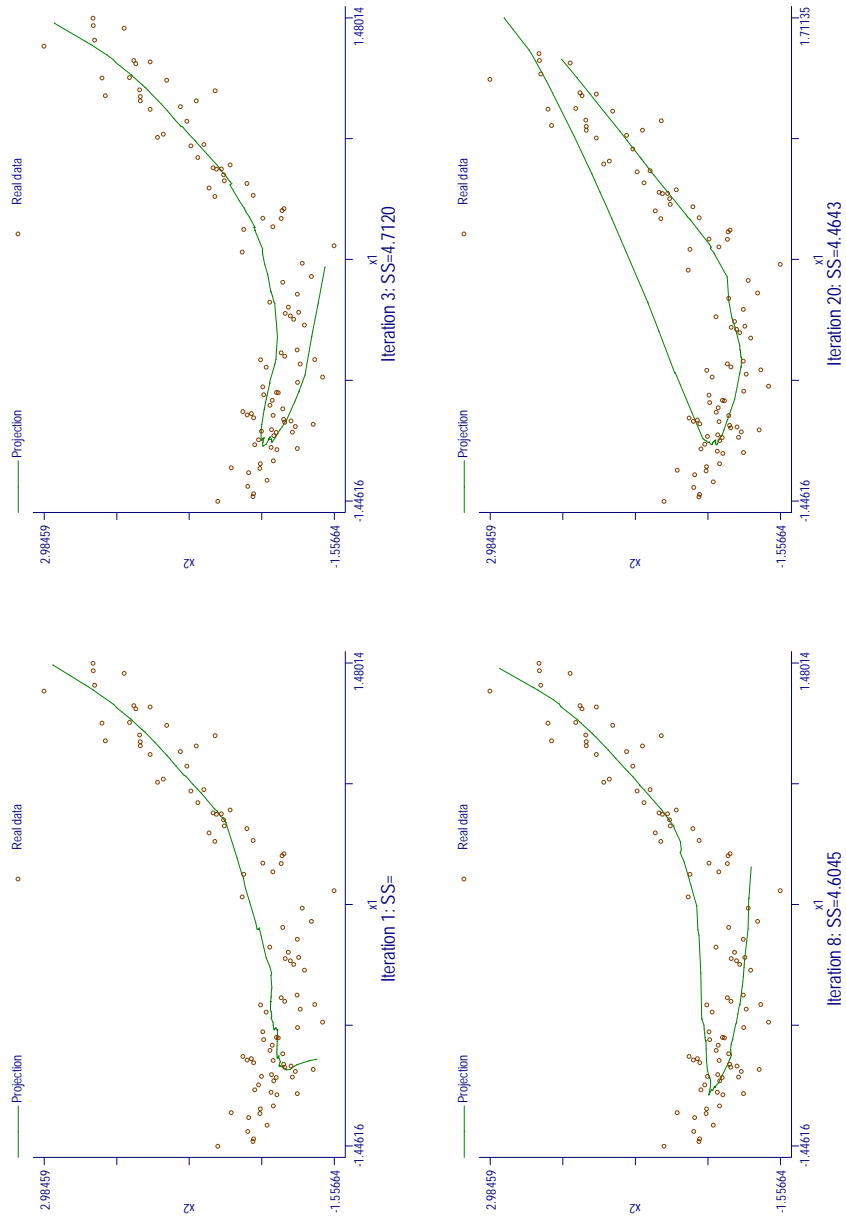


Figure 3: Parabola data — iterations



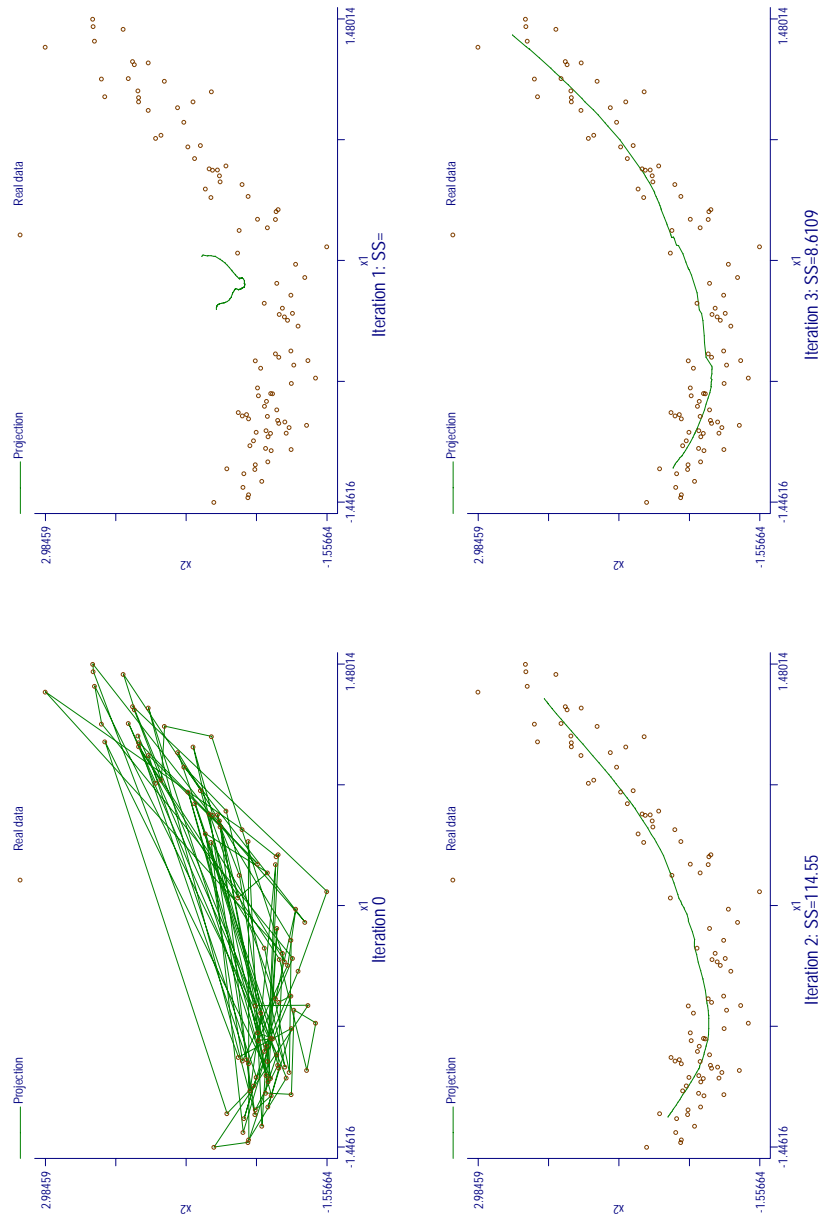
Parabola data

Figure 4: Parabola data — iterations (another random sample).



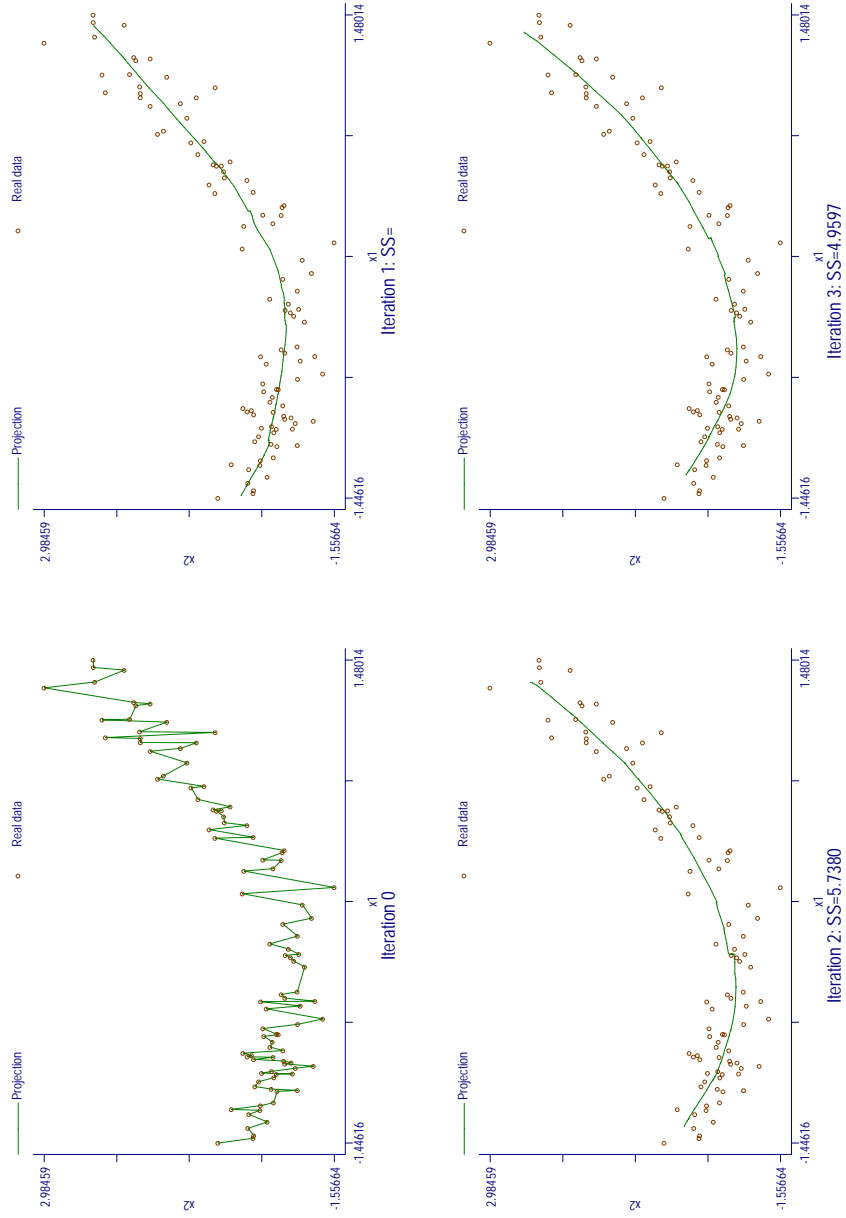
Parabola data

Figure 5: Parabola data — iterations; random starting point.



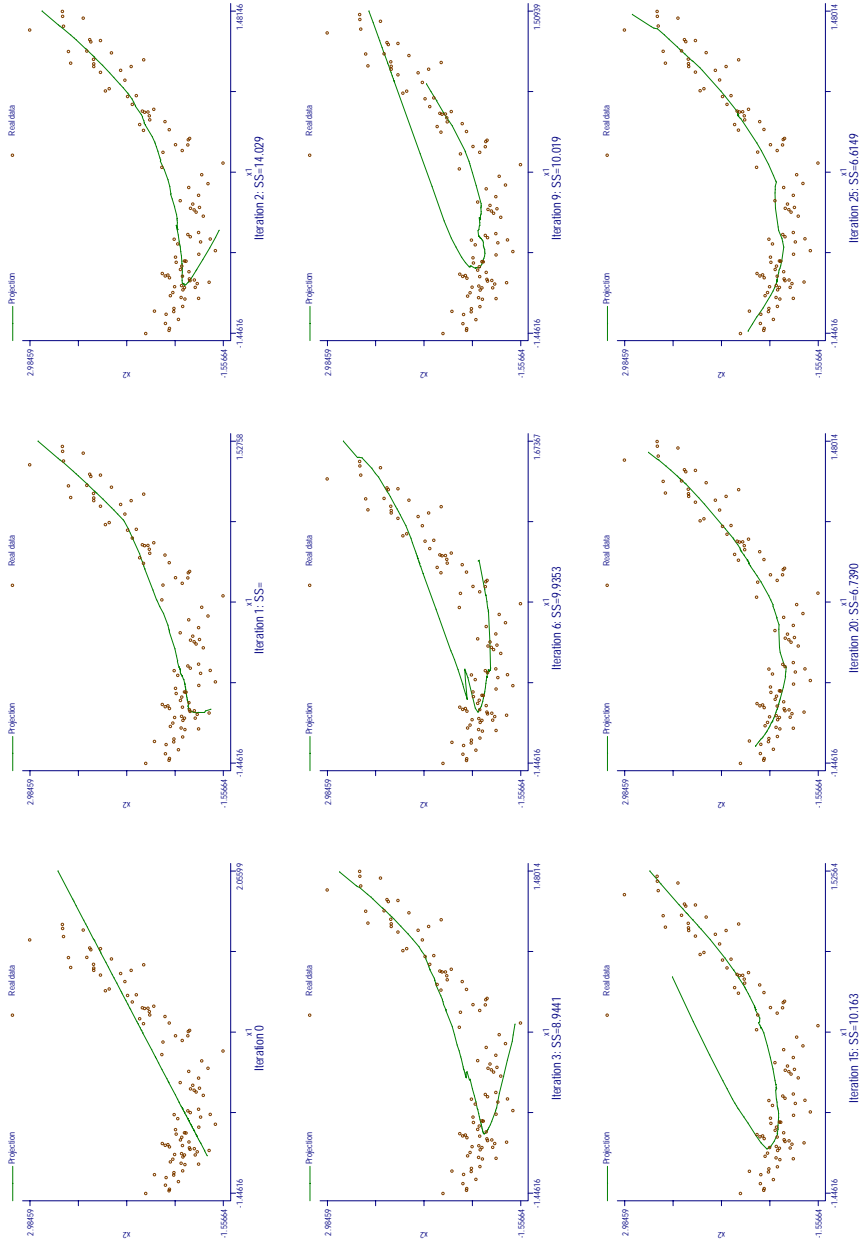
Circular data; seed = 4809; random starting point

Figure 6: Parabola data — iterations; starting at x1



Circular data; seed = 4809; start at x1

Figure 7: Parabola data — iterations; increased bandwidth.



Parabola data; seed = 4809; double bandwidth

Figure 8: Raw functional data.

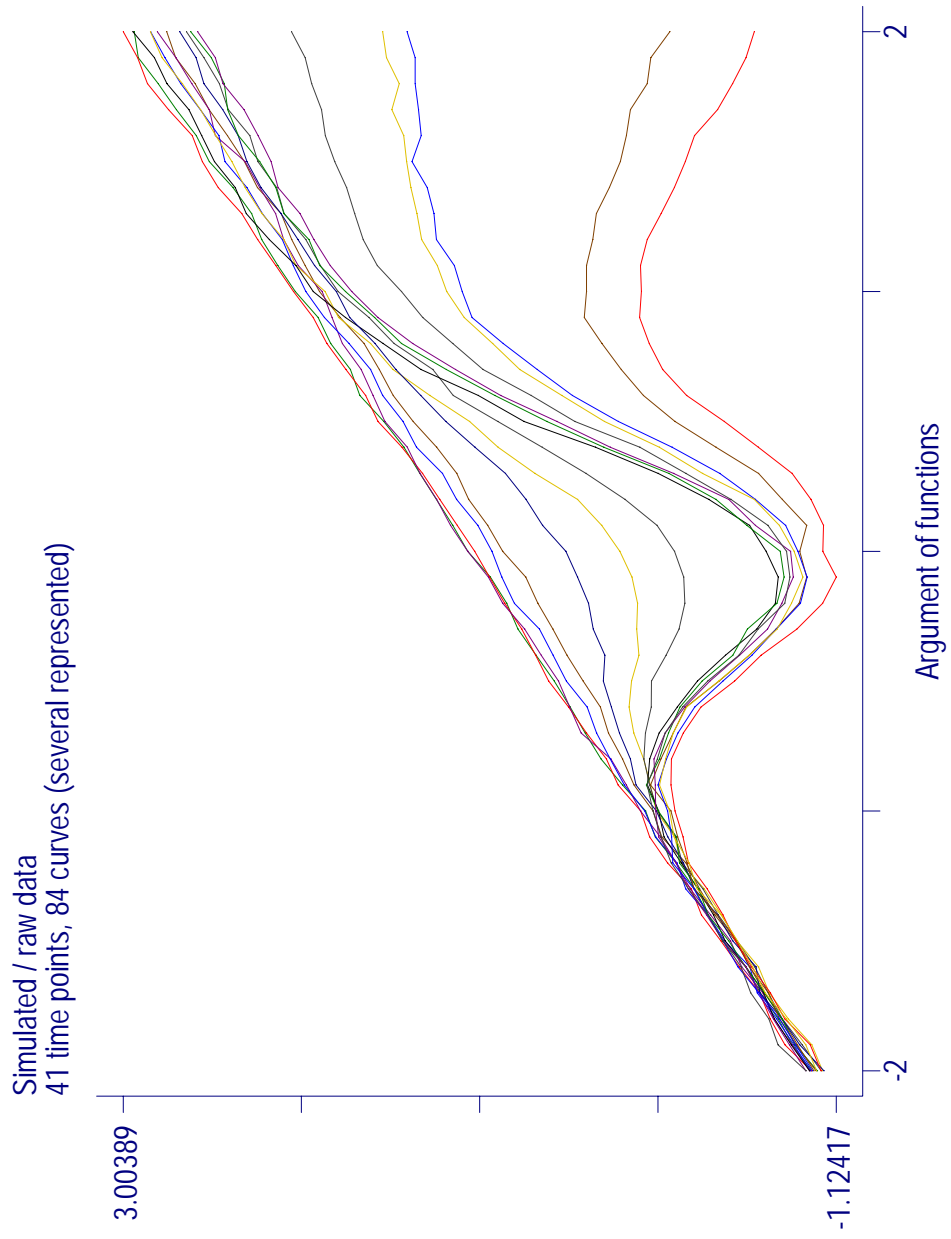


Figure 9: Principal components analysis of the data from Fig. 8.

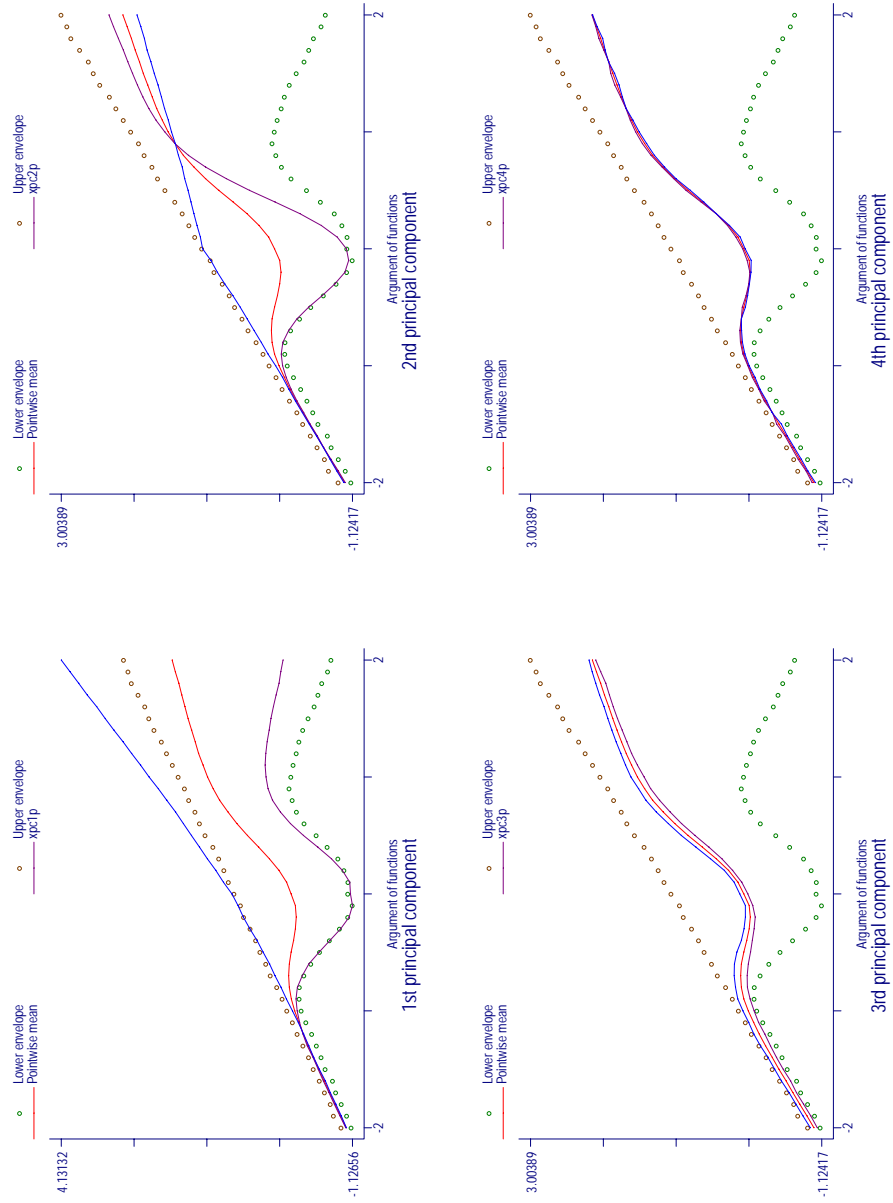
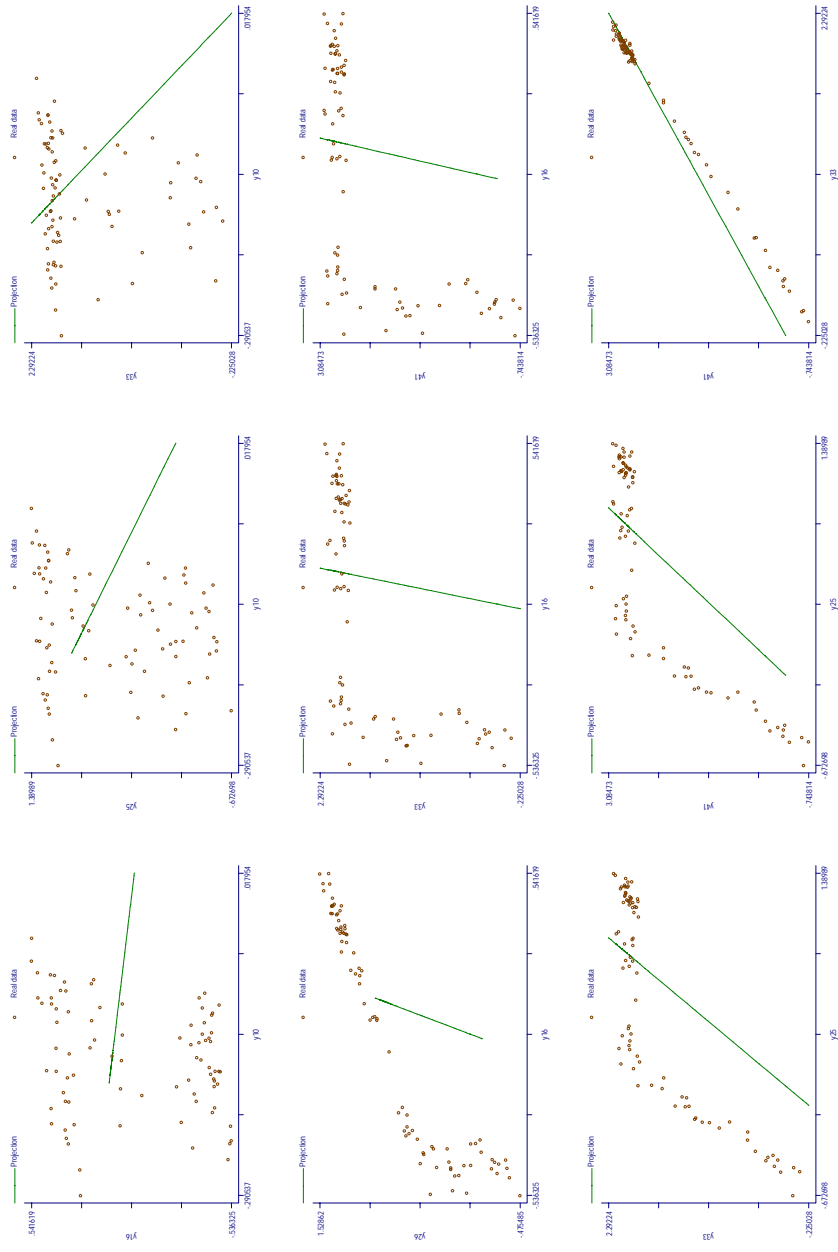
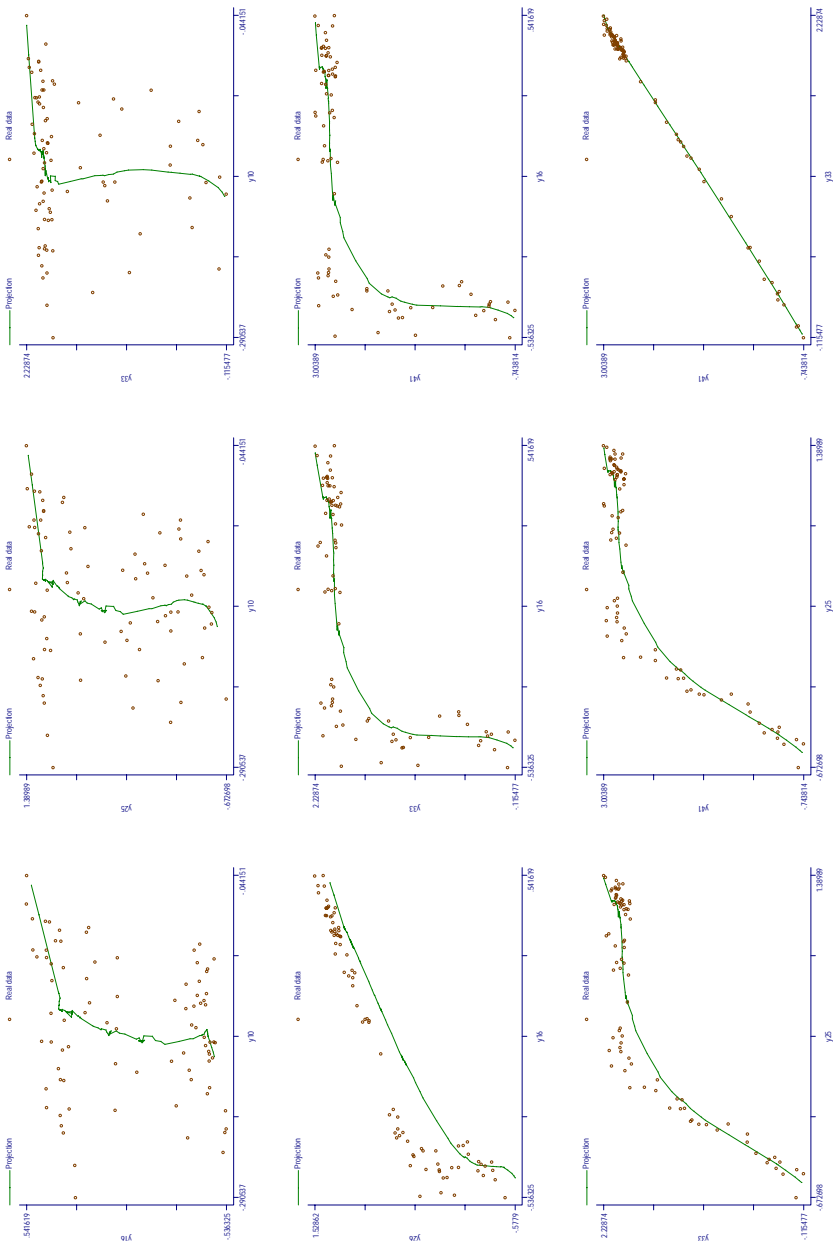


Figure 10: Iteration 0 with the curve data.



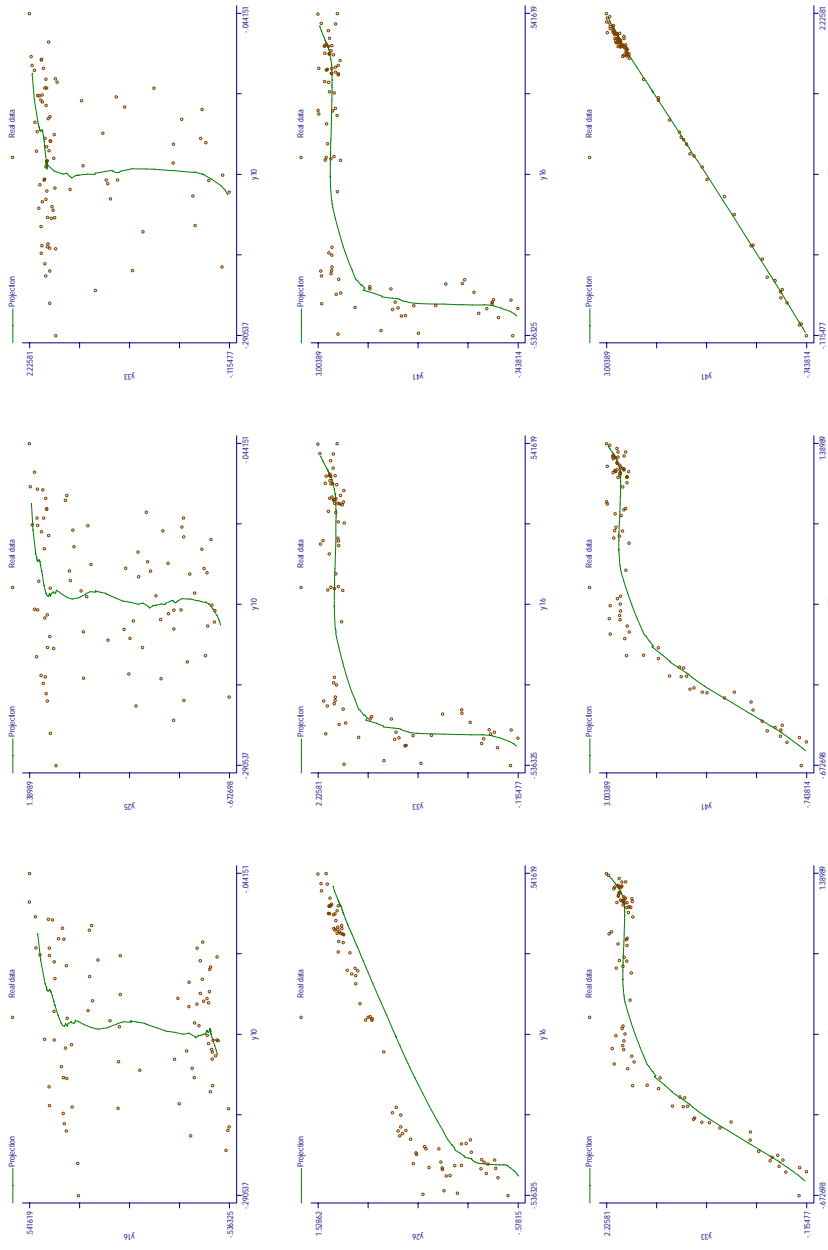
Iteration 0

Figure 11: Iteration 1 with the curve data.



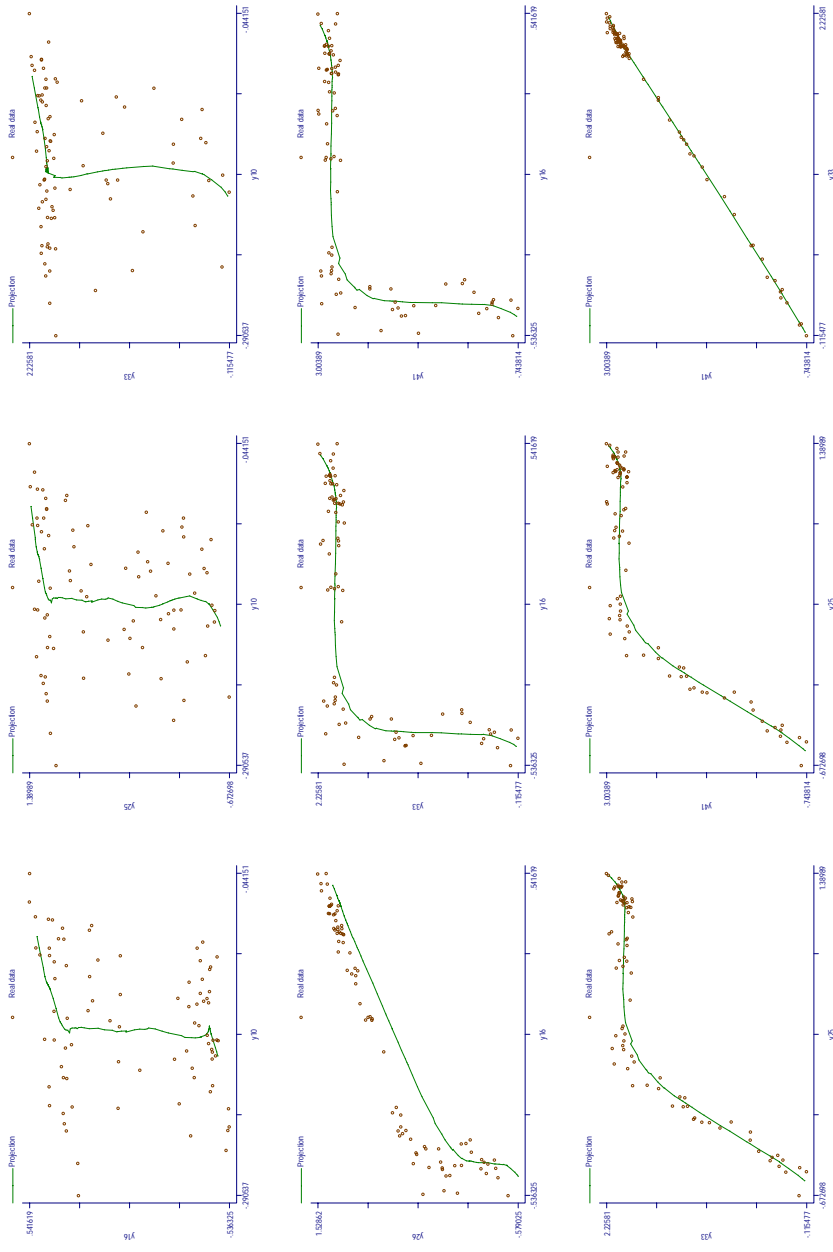
Iteration 1: SS=

Figure 12: Iteration 2 with the curve data.



Iteration 2:  $SS=28.637$

Figure 13: Iteration 3 with the curve data.



Iteration 3: SS=16.514

Figure 14: Projection of the data points onto the principal curve.

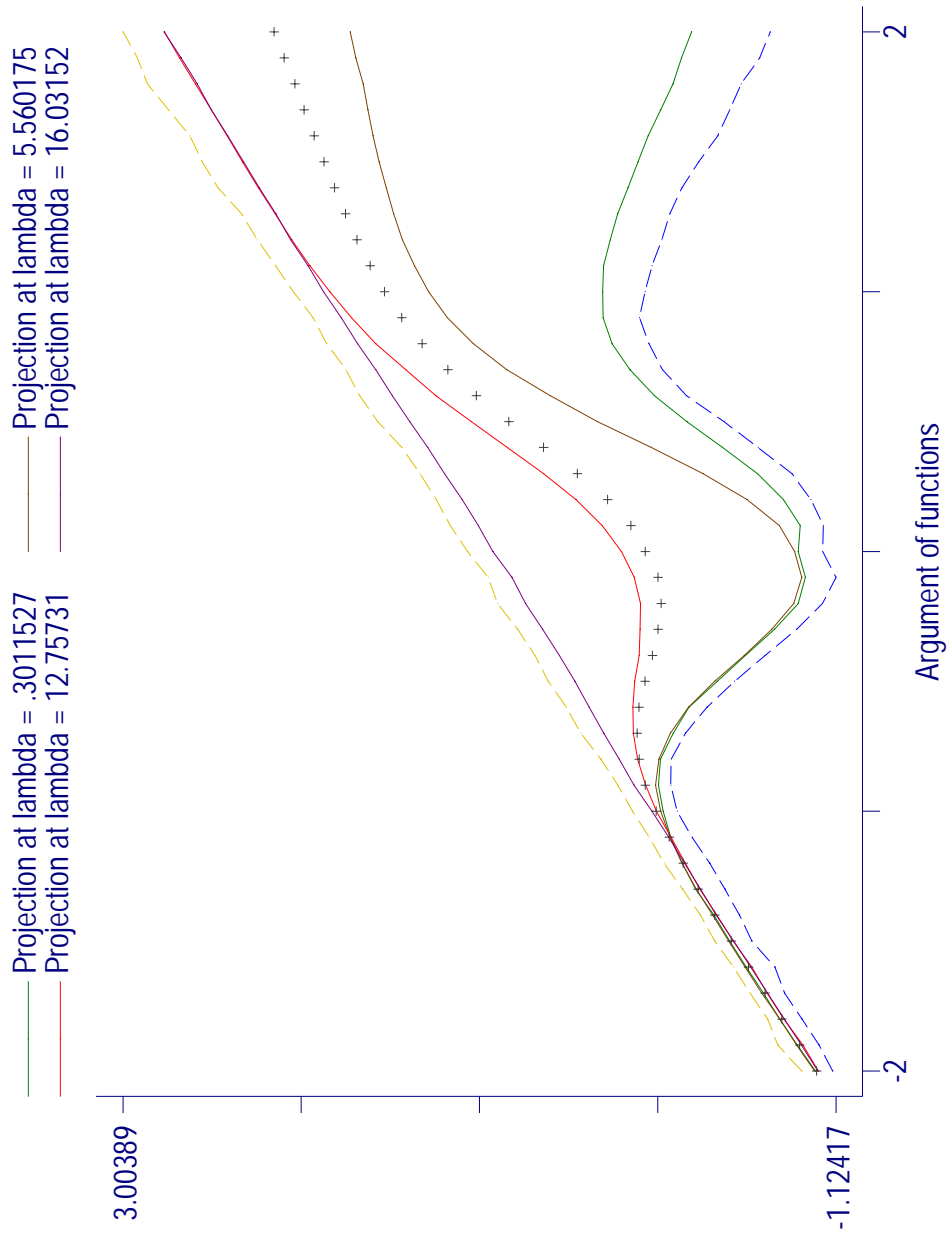


Figure 15: Parameters of the data generating process for the curves.

