Nonlinear time series analysis with the TISEAN package — Prediction, Modelling and Noise Reduction

Eckehard Olbrich

MPI MIS Leipzig

Potsdam SS 2008

Prediction and Modeling

Deterministic dynamics

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}(\boldsymbol{x}_{n+1})$$

which reduces to

$$x_{n+1} = F(x_n, \ldots, x_{n-m+1})$$

in the case of delay embedding.

Local methods: Basic idea: Looking for similar events in the past and using their future for prediction.
Local constant (*lzo-run*): Using the average as prediction.
Local linear (*lfo-run*): Fitting a linear model for similar events, i.e. neighbors in phase space.

Global models: Parameterizing the function $\boldsymbol{F}$ and fitting the parameters.
Polynomials (*polynom*)
Radial basis functions (*rbf*)
Neural networks (not included in TISEAN)

Local constant prediction — *lzo*

We want to predict x_{n+1}.

1. Looking for similar events in the past. Formally, looking for $\boldsymbol{x}_k$ with $|\boldsymbol{x}_n - \boldsymbol{x}_k| \leq \epsilon$, thus $\boldsymbol{x}_k \in \mathcal{U}_\epsilon(\boldsymbol{x}_n)$
2. Then our prediction is

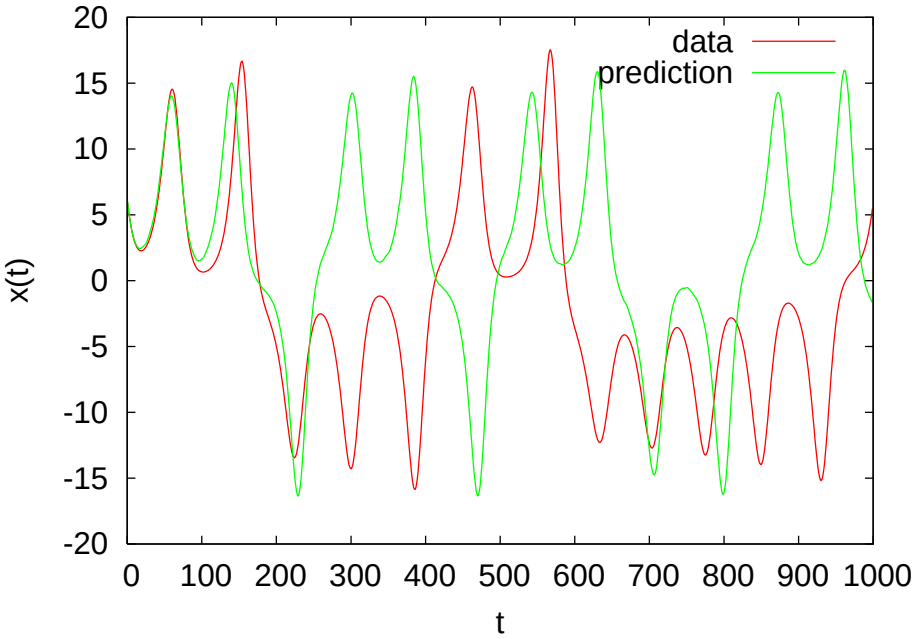
$$\hat{x}_{n+1} = \frac{1}{|\mathcal{U}_\epsilon(\boldsymbol{x}_n)|} \sum_{k:\boldsymbol{x}_k \in \mathcal{U}_\epsilon(\boldsymbol{x}_n)} x_{k+1}$$

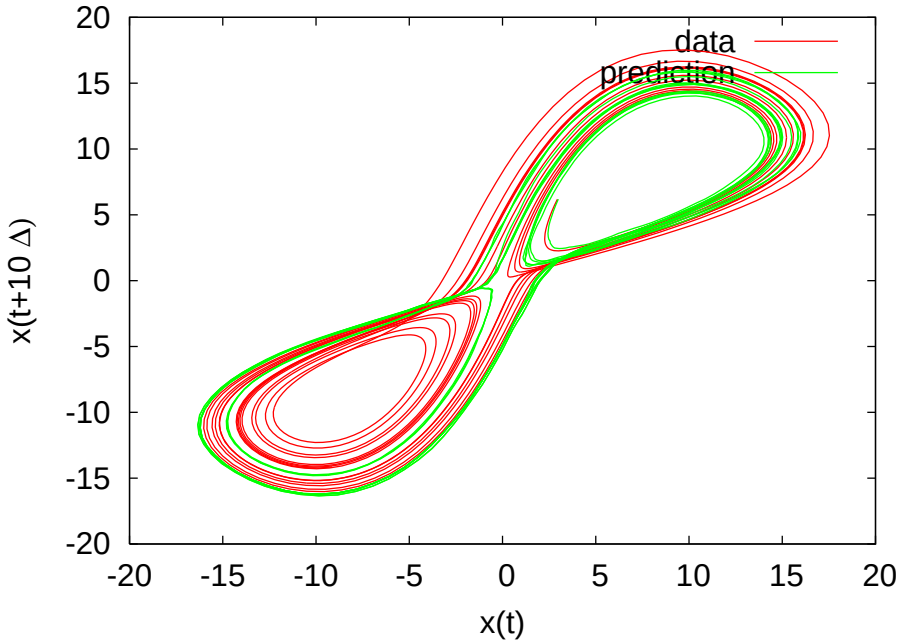
Parameter:

- Embedding parameter: delay d and embedding dimension m
- Minimal number of neighbours and/or neighbourhood size

Local constant prediction

Lorenz system, $m = 5$, $d = 5$





Advantage: High flexibility and robustness.

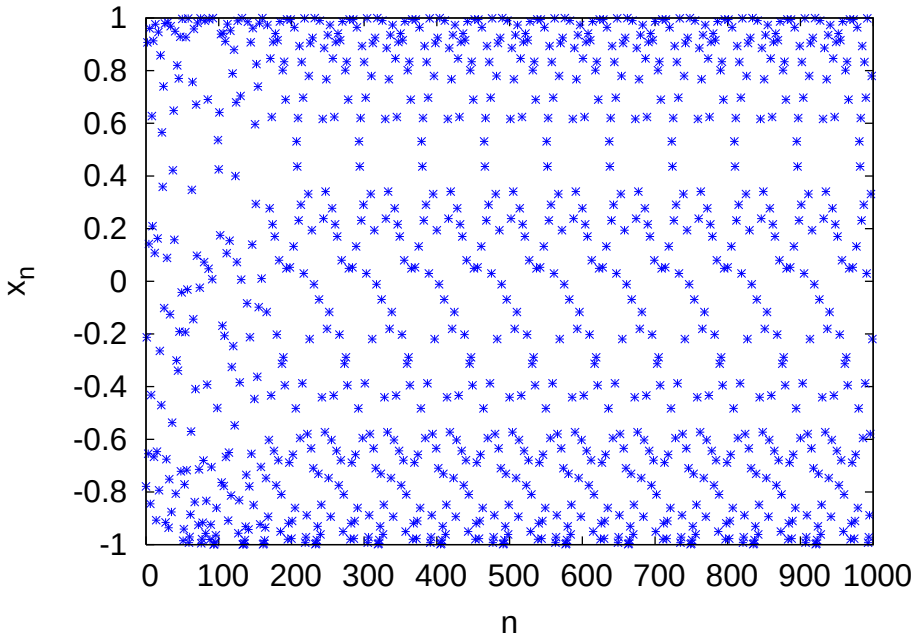
Disadvantage: Bad approximation of $\boldsymbol{F}(\boldsymbol{x}_n)$.

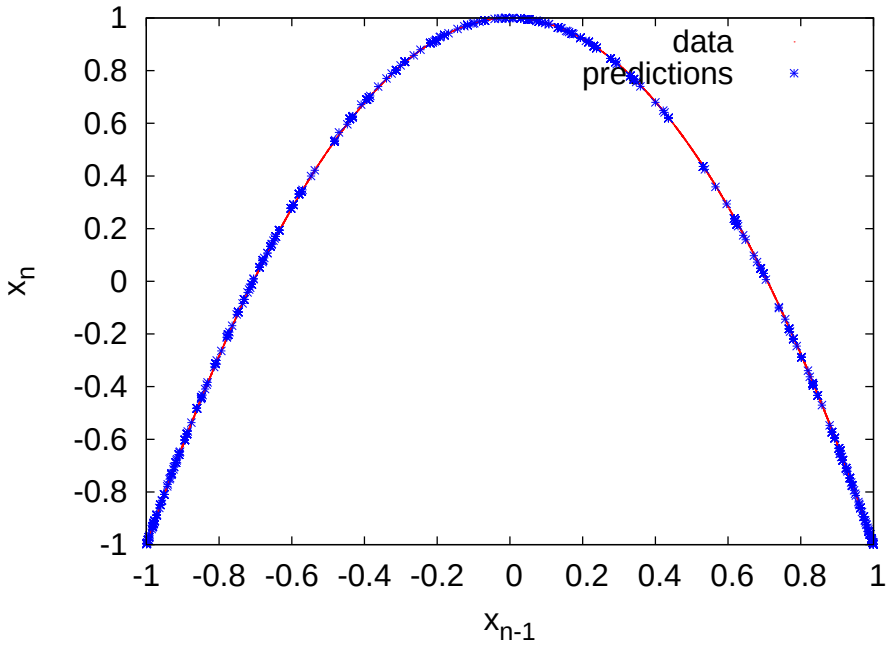
- Large bias at the boundaries.
- Dynamics more regular than the original one.

Optimal number of neighbours: trade-off between bias and variance of $\hat{\boldsymbol{F}}$

Local constant prediction

Logistic map, $m = 2$, default parameters





Advantage: High flexibility and robustness.

Disadvantage: Bad approximation of $\boldsymbol{F}(\boldsymbol{x}_n)$.

- Large bias at the boundaries.
- Dynamics more regular than the original one.

Optimal number of neighbours: trade-off between bias and variance of $\hat{\boldsymbol{F}}$

Local linear prediction — *lfo*

We want to predict x_{n+1}.

1. Looking for similar events in the past. Formally, looking for $\boldsymbol{x}_k$ with $|\boldsymbol{x}_n - \boldsymbol{x}_k| \leq \epsilon$, thus $\boldsymbol{x}_k \in \mathcal{U}_\epsilon(\boldsymbol{x}_n)$
2. Now our prediction is not the average of the futures of the similar from the past, but made by a linear model of these events.

$$\hat{x}_{n+1} = \boldsymbol{A}_n \boldsymbol{x}_n + \boldsymbol{b}_n$$

with

$$\hat{x}_{k+1} = \boldsymbol{A}_n \boldsymbol{x}_k + \boldsymbol{b}_n$$

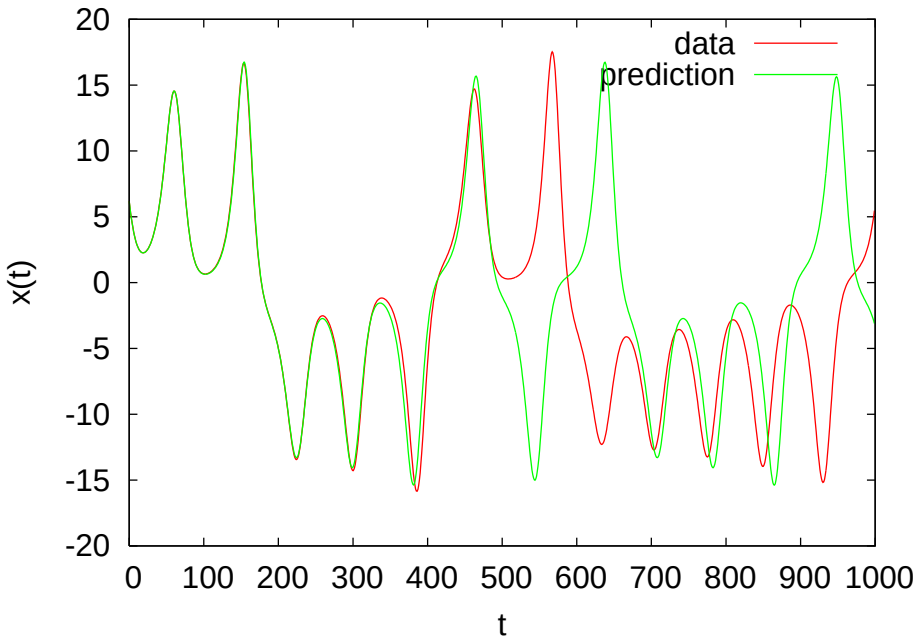
being the optimal linear predictor for the x_{k+1}.

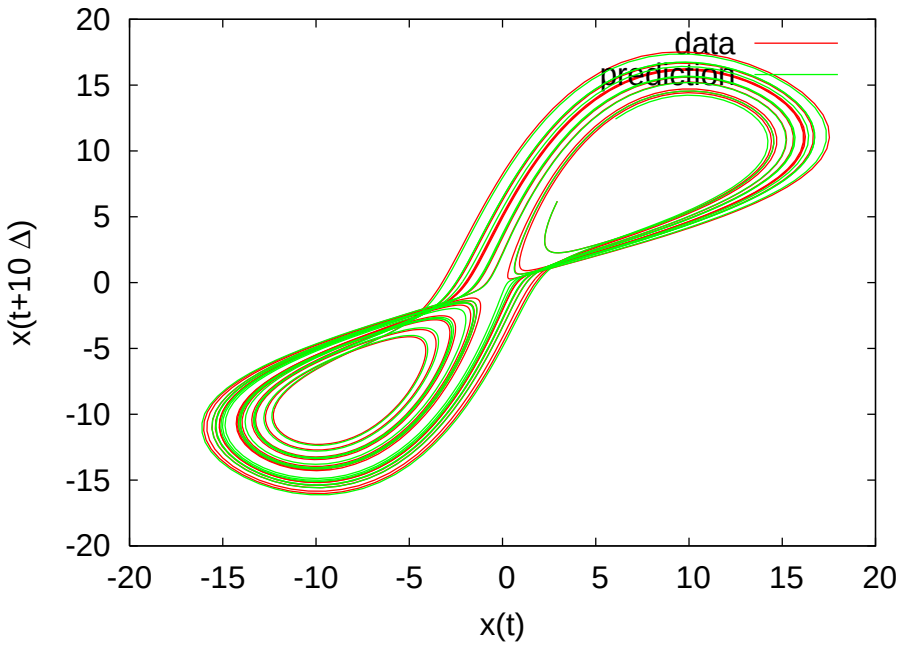
Parameter:

- Embedding parameter: delay d and embedding dimension m
- Minimal number of neighbours and/or neighbourhood size

Local linear prediction

Lorenz system, $m = 5$, $d = 5$





Advantage: High flexibility, better model than local constant model

Disadvantage: Less robust than local constant, in particular with noisy data

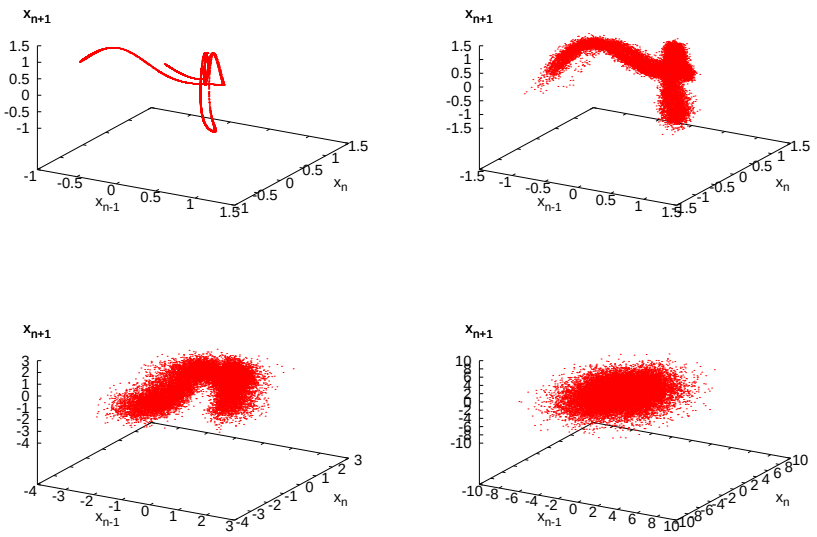
From nonlinear back to linear models - *lfo-ar*

Local linear models provide an easy way to check for nonlinear deterministic structure in the data by varying the neighbourhood size.

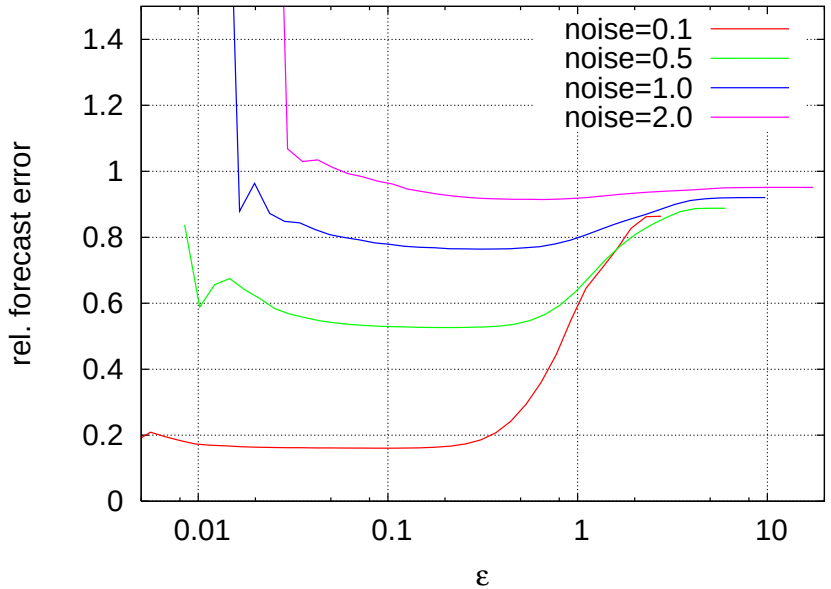
- Small neighbourhood size $\longrightarrow$ nonlinear structure is captured if present
- Increasing the neighbourhood size $\longrightarrow$ model converges to a global linear model, a AR(m)-model.
- A distinct minimum in the plot of the prediction error via the neighbourhood size indicates non-linearity.
- Note: We have to assume some dynamical noise on the data, otherwise the global linear model would produce a fixed point.

Example 2-d map with dynamical noise

$$x_{n+1} = 2(e^{-2x_n^2} - 1/2) + 0.3 \cdot x_{n-1} + \xi_n$$



Delay plots for different noise amplitudes



Relative prediction error vs. local neighbourhood size

Global models — Polynoms

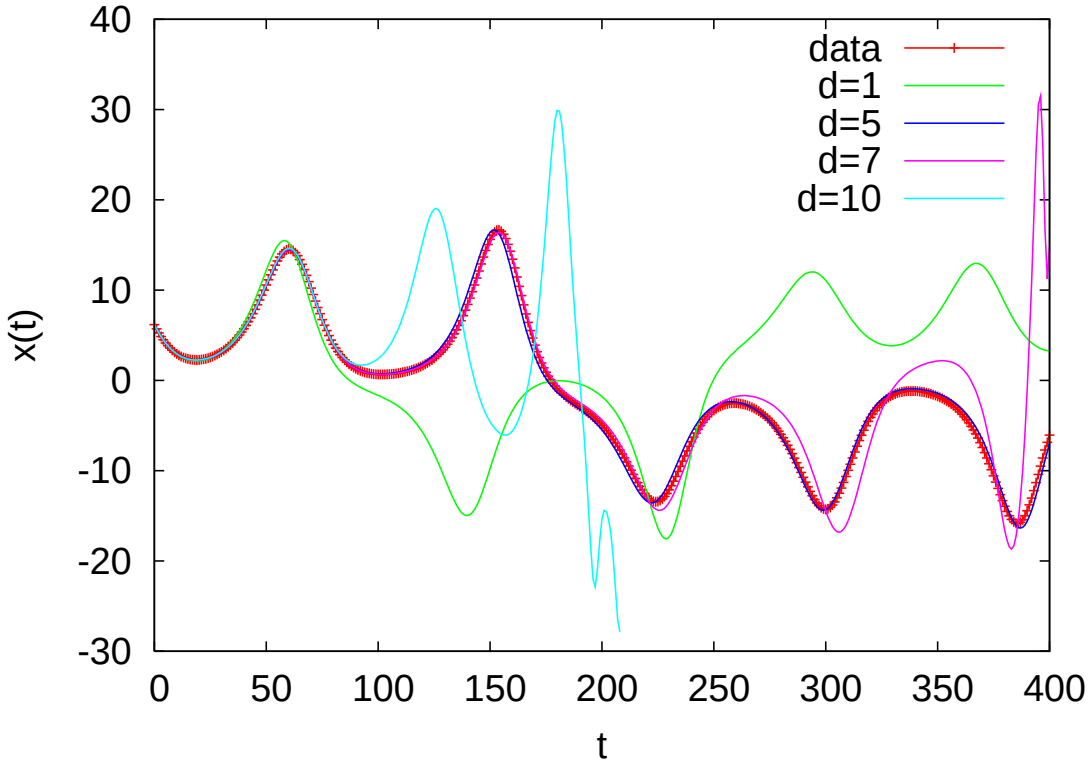
$$F(x) = \sum_{i_1,\ldots,i_m} a_{i_1 i_2 \ldots i_m} x_n^{i_1} x_{n-1}^{i_2} \cdots x_{n-m+1}^{i_m}$$

with the sum going over all m-tupel $(i_1, \ldots, i_m)$ with $\sum_{k=1}^{m} i_k \leq p$.
Parameter:

- Embedding parameter: delay d and embedding dimension m
- Order of the polynom p. $p = 1$ corresponds to a linear model.

Polynomial models - Example

Lorenz system: $m = 5,\ p = 3$



Forecast errors:

$d = 1$	1.153983e-04
$d = 5$	7.063427e-05
$d = 7$	6.116494e-05
$d = 10$	7.397461e-05

Polynomial models - Properties

Advantage: Easy to interprete (at least in some cases)

Disadvantage: Very often unstable — trajectory becomes unbounded, explosion of number of parameters

How to deal with the combinatorial explosion of parameters?

- Backward elimination: The program *polyback* removes term by term down to a given final number of remaining terms. The terms are removed in such a way that the onestep forecast error increases minimally.
- Note that this procedure might not find the optimal model of a certain size (i.e. number of parameters)

Global models — Radial Basis Functions

$$F(\boldsymbol{x}) = a_0 + \sum_{k=l}^{p} a_k \Phi(||\boldsymbol{x} - \boldsymbol{y}_i||)$$

- Functions $\Phi(r)$ are bellshaped, i.e. maximal at $r = 0$ and rapidly decaying towards zero with increasing r.
- Number and width of the functions Φ being fixed, the estimation of the a_k is a linear problem and could be estimated using least squares.
- rbf in TISEAN uses Gaussians with the standard deviation of the data

Some general remarks — Prediction

- Minimizing the root mean square error was a maximum likelihood estimator for Gaussian residuals. Here we have non-linear models with non-Gaussian residuals in general. Thus estimated predictors are not necessarily optimal.
- Also the problem of overfitting might occur. If one has enough data, one could use the out of sample error for model selection. Or one could use one of the information criteria (AIC, BIC, ...) although usually the underlying assumptions are not met.
- Conceptionally appealing is minimal description length (MDL) principle by Rissanen: The more parameters the model has the more bits are needed to ecode the model, but the less bits are needed for encoding the residuals. The best model is the model which leads to the shortest encoding of the data.

Some general remarks — Modelling

- Prediction $\neq$ Modelling. Minimizing the one-step prediction error leads usually to good short term predictions, but not to nessecarily to good models.
- The model might be

Some general remarks — Modelling

- Prediction $\neq$ Modelling. Minimizing the one-step prediction error leads usually to good short term predictions, but not to nessecarily to good models.
- The model might be
  - Unstable, i.e. the trajectory diverges (*polynom*).

Some general remarks — Modelling

- Prediction $\neq$ Modelling. Minimizing the one-step prediction error leads usually to good short term predictions, but not to nessecarily to good models.
- The model might be
  - Unstable, i.e. the trajectory diverges (*polynom*).
  - Converge to an attractor at different positions in the phase space.

Some general remarks — Modelling

- Prediction $\neq$ Modelling. Minimizing the one-step prediction error leads usually to good short term predictions, but not to nessecarily to good models.
- The model might be
  - Unstable, i.e. the trajectory diverges (*polynom*).
  - Converge to an attractor at different positions in the phase space.
  - Model shows qualitatively different behavior than the data, e.g. periodic instead of chaotic (*lzo-run*)
- No general scheme to find good models
- One possibility: minimizing n-step prediction errors instead of only the 1-step prediction error. Minimizing the n-step prediction error is, however, already a **non-linear** problem. Thus there are computational problems and also problems of existence and uniqueness.

How to evaluate your model?

- Prediction error characterizes only short term behaviour
- How to characterize the long term behaviour?
  - Estimate the dynamical invariants — dimension, entropies, Lyapunov exponents — from your model and directly from the data
  - For noisy data: compare statistical properties of data generated by the model with the original data, e.g. the correlation integral $C(m, \epsilon)$.

What if the systems are not perfectly deterministic?

We distinguish two kinds of **noise**:

Measurement noise: Here the dynamical system is still deterministic

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}(\boldsymbol{x}_n)$$

but observed via a noisy channel

$$\boldsymbol{y}_n = h(\boldsymbol{x}_n) + \boldsymbol{\nu}_n \ .$$

Dynamical noise:

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}(\boldsymbol{x}_n) + \boldsymbol{\xi}_n \ .$$

Here we have a qualitative different dynamics. The noise term can stand for high-dimensional and/or high-entropic processes. In the following we will consider only the case of measurement noise.

Noise reduction

In the case of measurement noise there is a well defined noiseless trajectory. Noise reduction means removing the noise from the data by applying some filter to the data. In TISEAN there are several possibilities implemented:

- Wiener filter (*wiener*)
- Savitzky-Golay filter (*sav_gol*)
- Simple nonlinear noise reduction (*lazy*)
- Projective nonlinear noise reduction (*ghkss*)

The first two are linear filters, the last two non-linear ones.

Wiener filter

Idea: contribution fo the system and of the noise are well separated in the spectrum or are at least additive.

$$y(f_k) = x(f_k) + \nu(f_k)$$

$y(f_k)$, $x(f_k)$ and $\nu(f_k)$ are the Fourier transforms of x_n, y_n and ν_n, respectively.
The Wiener filter ϕ_k: $x(f_k) \approx \phi_k y(f_k)$ should minimze

$$\begin{aligned} e^2 &= \sum_k (\phi_k y(f_k) - x(f_k))^2 \\ &= \sum_k (\phi_k - 1)^2 |x(f_k)|^2 + \phi_k^2 |\nu(f_k)|^2 \end{aligned}$$

Thus the Wiener filter becomes

$$\phi_k = \frac{x(f_k)}{y(f_k)} \, .$$

Wiener filter

Idea: contribution fo the system and of the noise are well separated in the spectrum or are at least additive.

$$y(f_k) = x(f_k) + \nu(f_k)$$

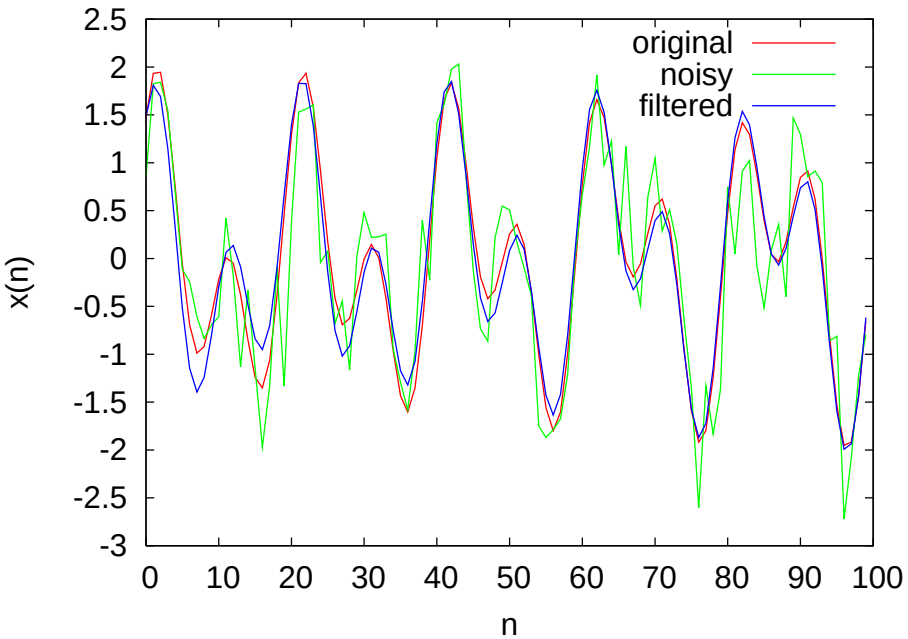
$y(f_k)$, $x(f_k)$ and $\nu(f_k)$ are the Fourier transforms of x_n, y_n and ν_n, respectively.
Thus the Wiener filter becomes

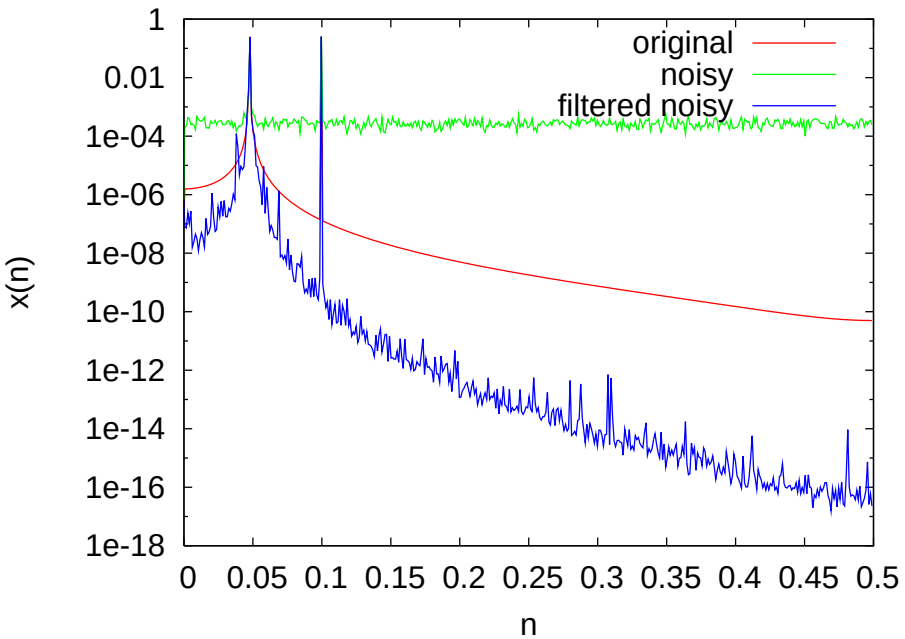
$$\phi_k = \frac{x(f_k)}{y(f_k)} \, .$$

Problems: If both system and noise have have considerably power at the same frequencies it is not possible to remove the noise by only adjusting the Fourier amplitudes as it is usually done.

Example: Quasiperiodic signal with additive noise

Data and power spectrum of the noise free, noisy and filtered signal.





Program *Wiener1* estimates $y(f_k)$. *Wiener2* needs some guess for $x(f_k)$ as input and produces then a cleaned time series $\hat{x}_n$

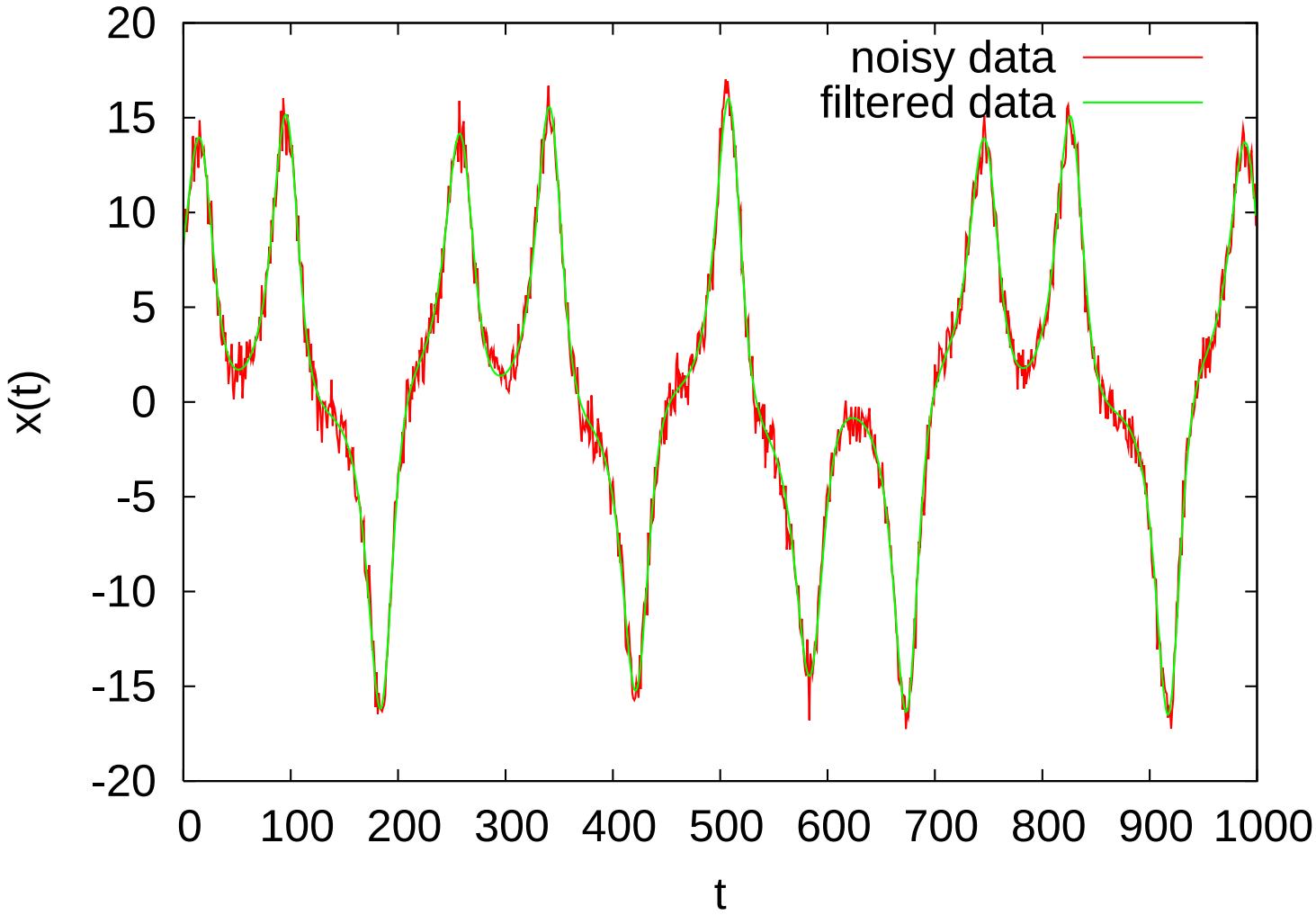
Savitzky-Golay filter — *sav_gol*

- Smoothing filter: It fits the trajectory (*nb* points in backward, *nf* points in forward direction) locally by a polynomial of order *p*.
- Removes high-frequency contributions
- In particular desgined to estimate temporal derivation on noisy data (option *D* gives the order of the derivative)
- Generally it has the form of a moving average filter

$$x_n = \sum_{k=n-nb}^{n+nf} a_k y_k$$

Savitzky-Golay filter — *sav_gol*

Example: Lorenz data with additive noise



Nonlinear noise reduction

- Both the Wiener and the Savitzky-Golay filter were linear filters and they often fail in removing the noise in nonlinear systems.
- In nonlinear deterministic systems we can use the underlying deterministic structure.

Embedding for noisy data

- While the embedding theorem does not apply for the noisy data $\{y_n\}$, it would apply for the noiseless observations $\tilde{x}_n = h(\boldsymbol{x}_n)$.
- Thus we can assume a deterministic dynamics $\tilde{\boldsymbol{F}}$ for the states $\tilde{\boldsymbol{x}}_n = (\tilde{x}_n, \ldots, \tilde{x}_{n-m+1})$ constructed by delay embedding from the noiseless observations.
- In the following we identify this delay embedding with the original state space.

Simple nonlinear noise reduction — *lazy*

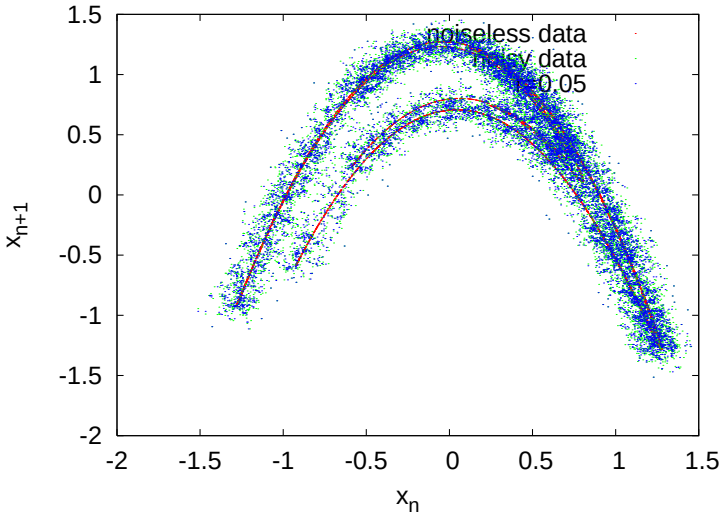
- Let us consider the delay vectors $\boldsymbol{y}_n = (y_n, \ldots, y_{n-m+1})$. $\mathcal{U}_\epsilon(\boldsymbol{y}_{n_0})$ is the set of all points in the ϵ-neighbourhood of $\boldsymbol{y}_{n_0}$.

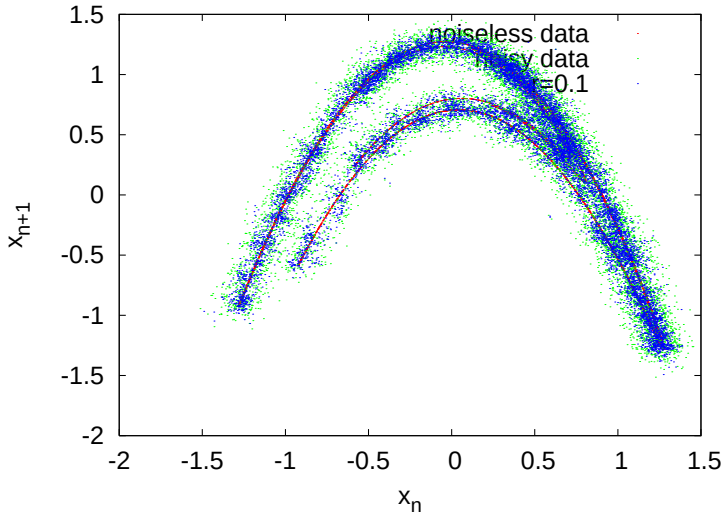
$$\hat{x}_{n_0-m/2} = \frac{1}{|\mathcal{U}_\epsilon(\boldsymbol{y}_{n_0})|} \sum_{\boldsymbol{y}_n \in \mathcal{U}_\epsilon(\boldsymbol{y}_{n_0})} y_{n_0-m/2}$$

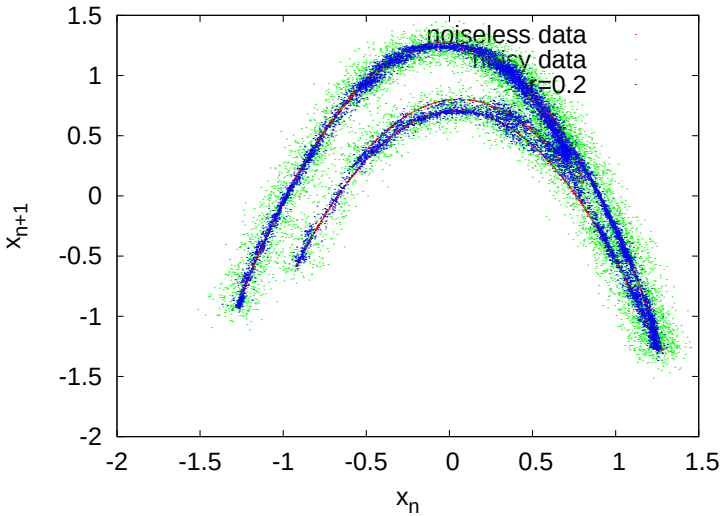
- Basically it uses the same procedure as the local constant prediction (*lzo-run*) method, but not for the next point in time, but for the point in the middle of the delay vector.
- Advantages and disadvantages of this algorithm for prediction apply: robustness and flexibility via. biased estimation for finite data sets.

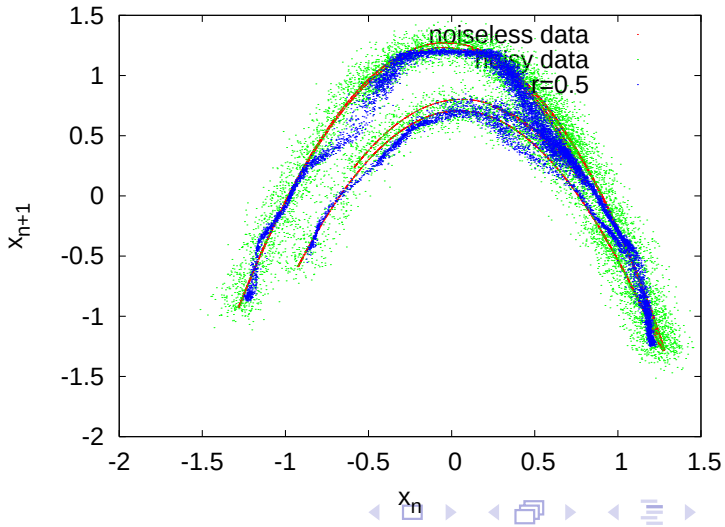
Example — Henon map with additive noise

lazy with different neighbourhood sizes r







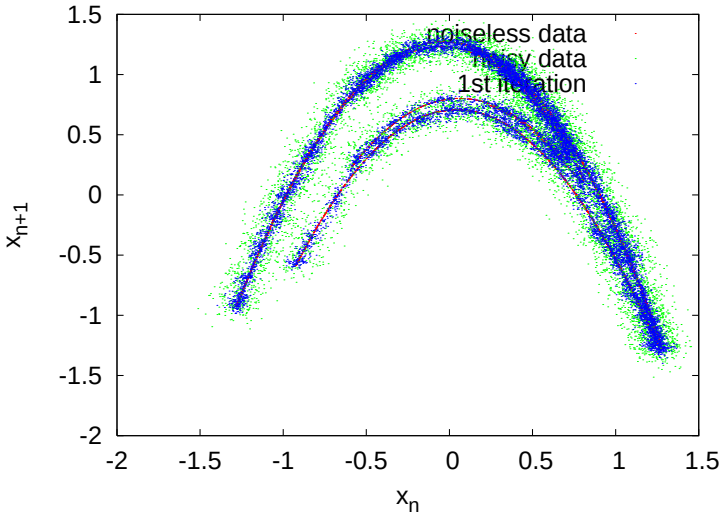


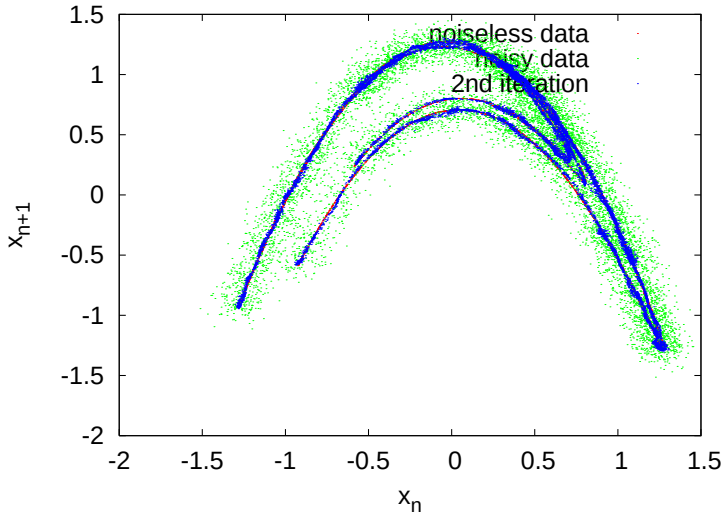
Projective nonlinear noise reduction

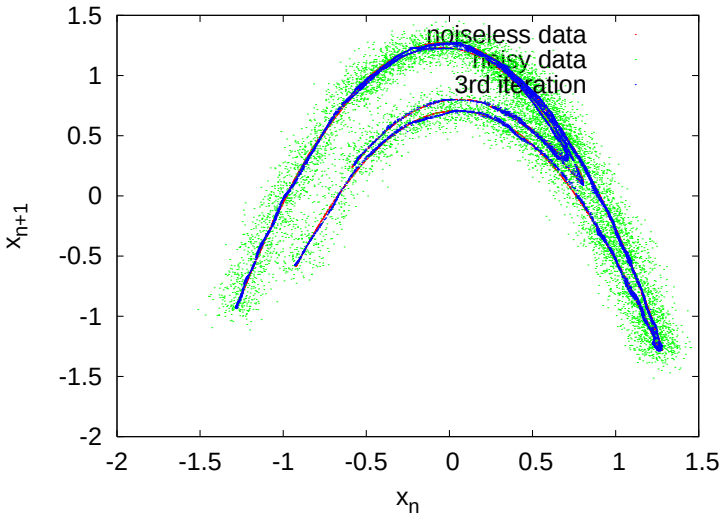
- Here we try to reconstruct the lower dimensional manifold, in which the deterministic trajectory is lying and project the noisy data points on this manifold.
- The attractor is locally approximated by a hyperplane. So this method corresponds to the local linear model.
- First published by Kostelich and Yorke (1988).
- TISEAN: *ghkss* implements the method proposed by **G**rassberger,**H**egger,**K**antz, Schaffrath and **S**chreiber, Chaos, 3(1994), 127.

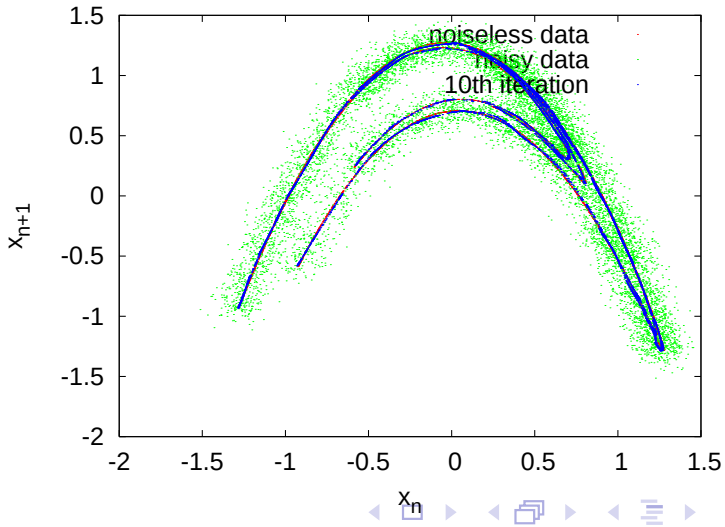
Example — Henon map with additive noise

1st,2nd,3rd and 10th iteration of *ghkss*



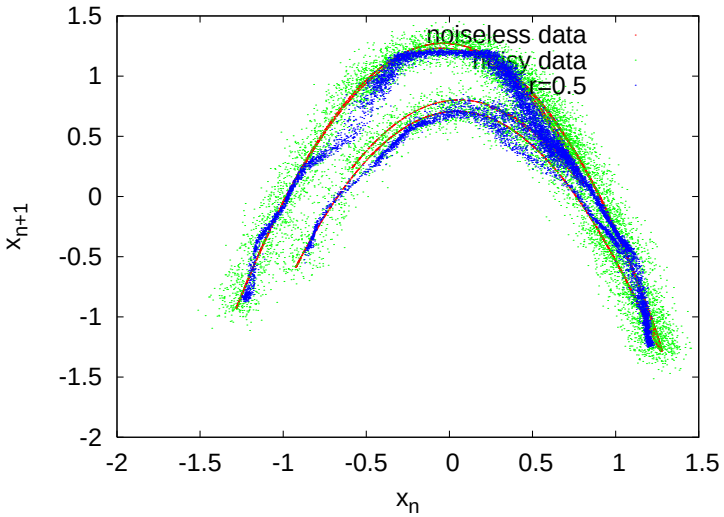






Simple noise reduction vs. locally projective noise reduction

lazy



ghkss