Week 7

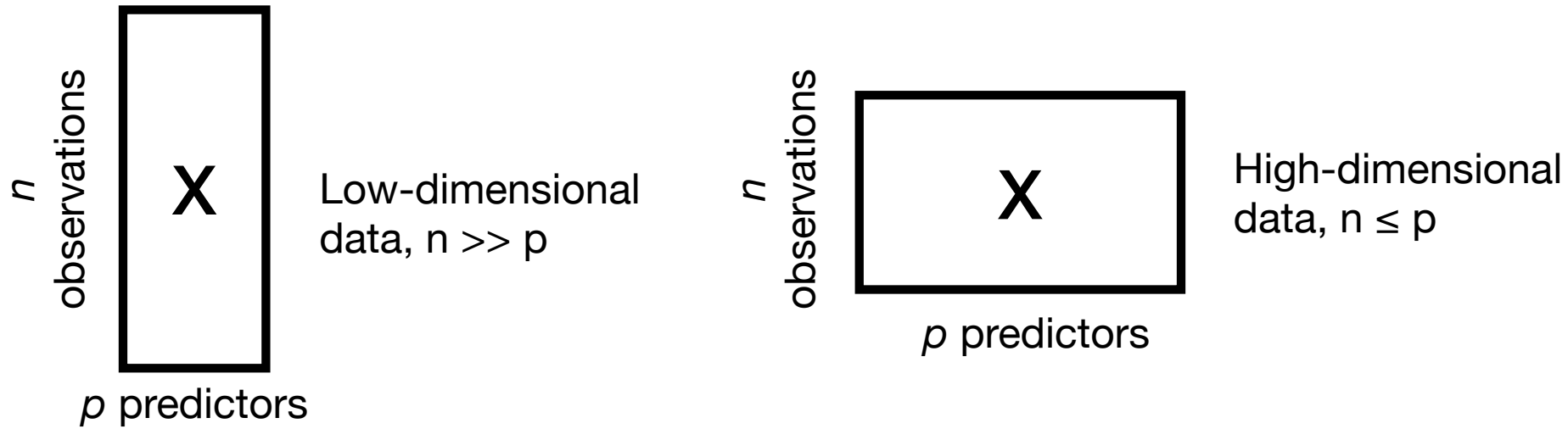
Cogs 109: Data Analysis and Modeling

Fall 2017

Prof. Eran Mukamel

Data dimensionality

- When we fit a model with *p* predictors, we have to search for the parameters in a p-dimensional space
- Each dimension is a “degree of freedom”
- When the number of predictors becomes very large (high-dimensional data), overfitting becomes more likely
- When $p \geq n$, there are more parameters than data points and we are in the high-dimensional regime



Data dimensionality

columns =
samples (~3,000)

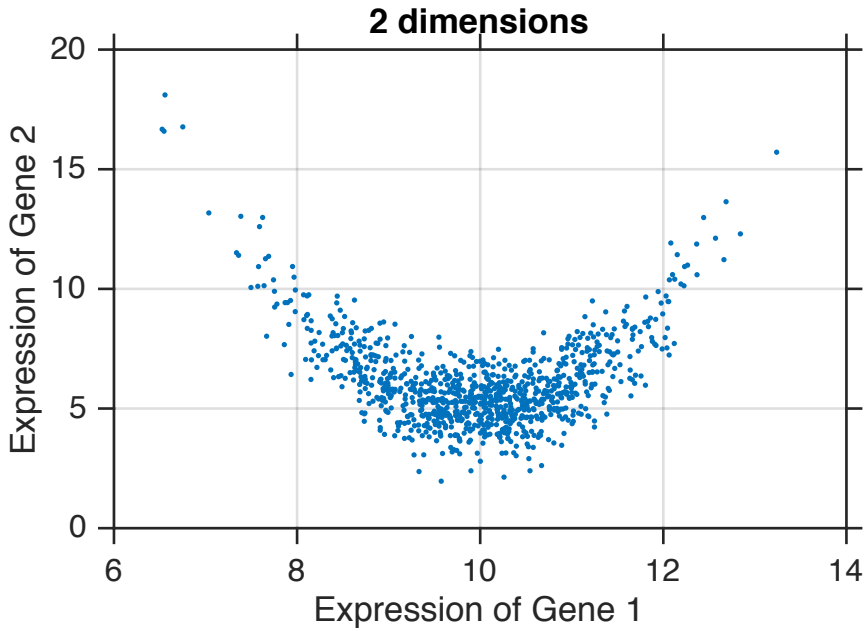
rows =
genes
(~25,000)

Variables – gene_exp

gene_exp

25000x100 double

	1	2	3	4	5	6
1	0.5377	1.3932	-1.5805	-0.5524	0.8712	0.5
2	1.8339	-1.2653	1.5955	0.1552	-1.2956	-0.2
3	-2.2588	0.0067	0.5043	2.2003	0.0313	1.0
4	0.8622	0.5153	0.6888	-1.8967	-0.9744	1.1
5	0.3188	-1.6947	0.0926	-0.1048	-1.5486	-0.1
6	-1.3077	1.1032	0.7451	0.5117	-0.0828	0.6
7	-0.4336	0.3182	-1.6702	0.6651	-0.6110	-0.4
8	0.3426	2.4211	1.2258	0.7907	0.2588	0.8
9	3.5784	0.8455	-0.5709	0.2087	-0.7201	0.1
10	2.7694	0.7924	-0.1149	0.4817	0.1704	-0.1
11	-1.3499	1.5456	-0.9211	0.6046	0.6634	0.4
12	3.0349	-0.3068	0.3893	0.3256	-1.2162	0.4
13	0.7254	1.1750	0.5140	-0.0172	0.3203	-0.6
14	-0.0631	-0.1419	0.1162	-0.3527	1.1505	-0.1
15	0.7147	0.5879	0.6677	0.2406	0.7859	0.1
16	-0.2050	0.1509	-0.8069	0.7951	0.7733	0.4
17	-0.1241	-0.2849	-0.9206	-0.9062	-2.9341	1.2
18	1.4897	-0.9960	-0.7019	1.0280	-1.1590	-0.2
19	1.4090	9.2824e-...	0.1699	-0.0913	0.3399	1.4
20	1.4172	0.1146	-1.8062	-1.3306	-0.7442	0.2
21	0.6715	0.0466	1.3939	-1.6348	-0.2118	-0.4
22	-1.2075	0.0263	-1.2782	-0.8169	1.2511	0.6
23	0.7172	-0.2156	-0.0260	-1.0981	-1.1597	0.8
24	1.6302	0.5771	1.0744	-0.0498	0.1236	0.1
25	0.4889	0.7458	-1.4722	-0.5208	-0.0863	-1.4
26	1.0347	0.8465	-2.4049	0.7310	-0.0806	0.1
27	0.7269	-0.0217	-1.0632	-0.1361	-0.4153	2.1



Dimensional reduction

- Subset selection (stepwise, best subset, etc.) deal with high-dimensional data by removing predictors
- Regularization (Ridge regression, LASSO) deals with the problem by shrinkage — constraining parameter values
- A third approach: Reduce data dimensionality by projecting onto a low-dimensional space

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

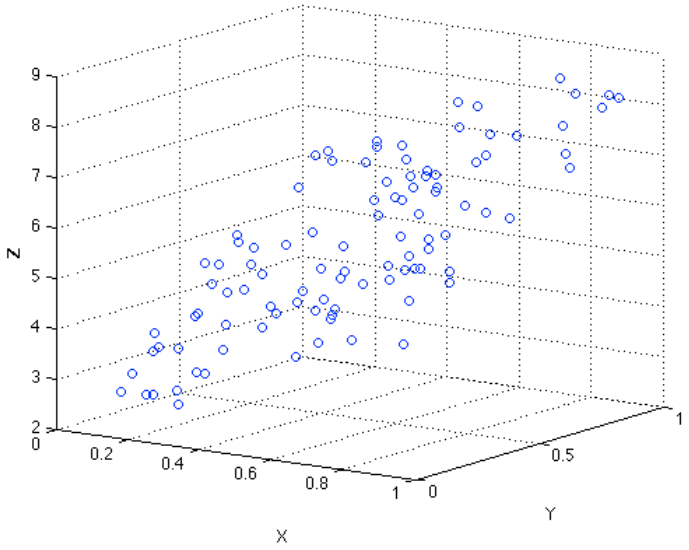
Lower-dimensional projection

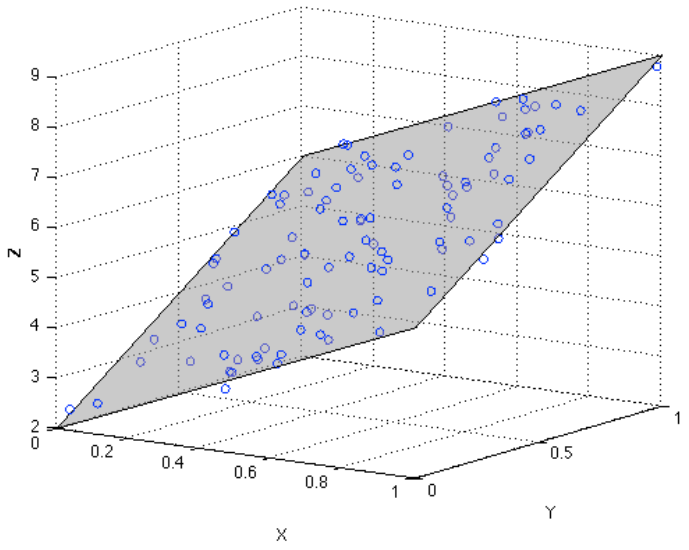
Projection weights

Predictors

Dimensionality reduction by PCA

- Principal components analysis (PCA) projects high-dimensional data onto a smaller number of “most interesting” dimensions





Defining the principal components (PCs)

- Principal components are linear combinations of the original predictors
- They are defined by:
    - All PCs are uncorrelated with each other
    - The first PC (PC1) has the maximum variance of any linear combination of the predictors
    - PC2 has the max. variance of any linear combination of predictors that is uncorrelated with PC1
    - PC3 has the max. variance of any linear combination of predictors that is uncorrelated with PC1 and PC2
    - Etc.

PC regression

- If we include all the PCs in our regression, the result will be exactly the same as using the original predictors
- If we include only the top k PCs ($k < p$) then we will be reducing the dimension of the predictors
- Since the top few PCs contain most of the data variance, they may be sufficient to predict the outcome (y)
- We assume that the remaining (p-k) dimensions contain mostly noise, and we exclude them

Regression in reduced-dimension

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i,$$

- Note that a linear combination of the principal components (z) is also a linear combination of the original predictors (x)

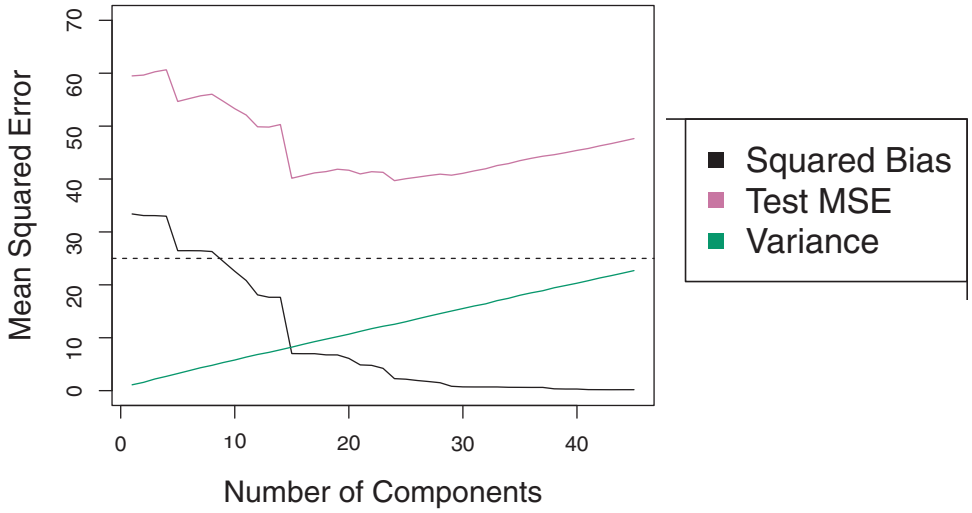
$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{jm} x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \phi_{jm} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij},$$

- Here, the value of beta is:

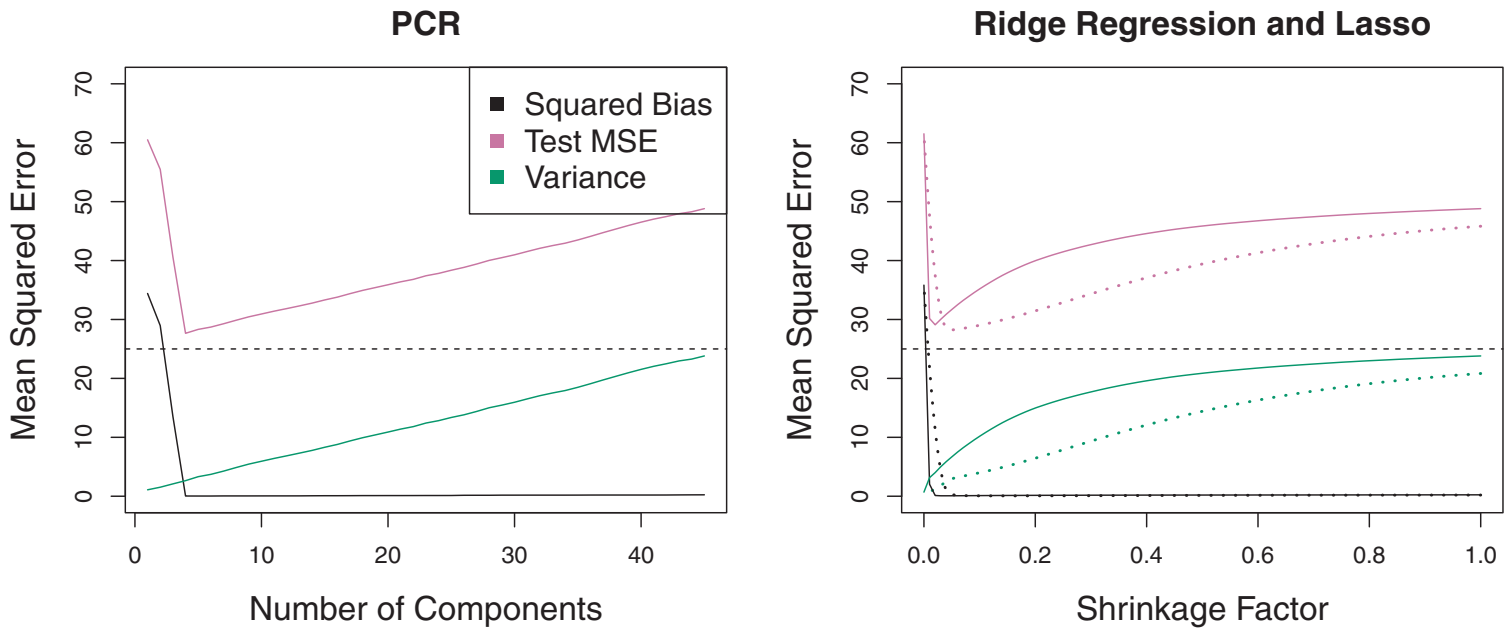
$$\beta_j = \sum_{m=1}^{M} \theta_m \phi_{jm}.$$

Model selection using PC regression

- As in subset selection and regularization, we need to choose a level of model complexity
- In PC regression, we do this by choosing how many PCs to keep (k)
- The procedure for selecting k is the same as in subset selection: Try each value, compute cross-validated MSEtest, and select the k value with the lowest MSEtest



Comparing PC regression with regularization



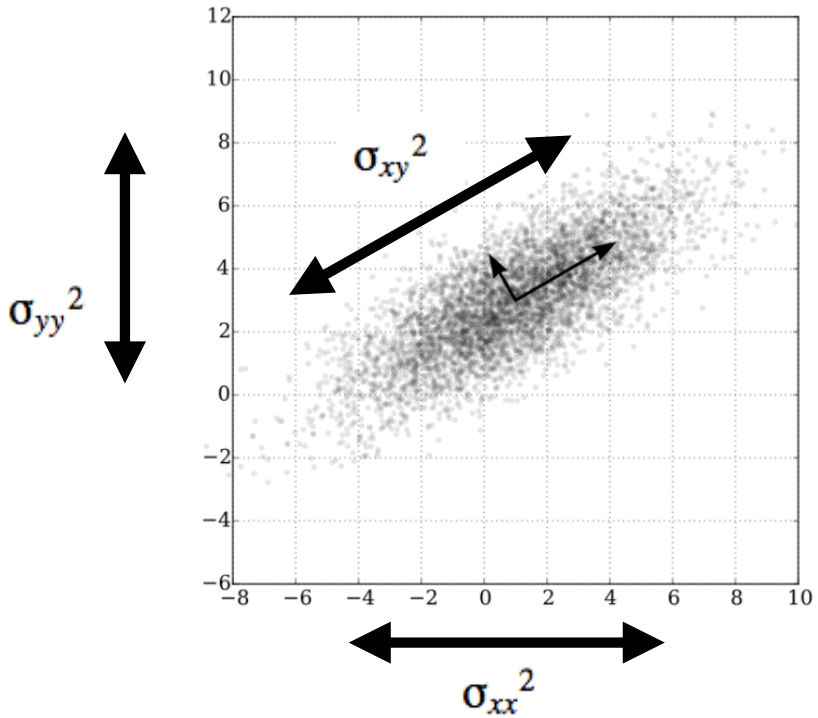
PC regression is not sparse

- LASSO and subset selection remove unneeded predictors. This is called a “sparse” model.
- Ridge regression uses all predictors — not sparse
- PC regression uses all predictors as well — not sparse

How does PCA work?

- Principal components are the eigenvectors of the *covariance matrix*

$$C_{ij} = \sigma_{ij}^2 = \frac{1}{N} \sum_k (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)$$



Non-linear regression methods

- Linear regression with one predictor:

$$y = \beta_0 + \beta_1 x$$

- Polynomial regression:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots = \sum_{k=0}^{K} \beta_k x^k$$

- Spline regression:

$$y = f(x, \beta_0, \beta_1, \ldots, \beta_K)$$

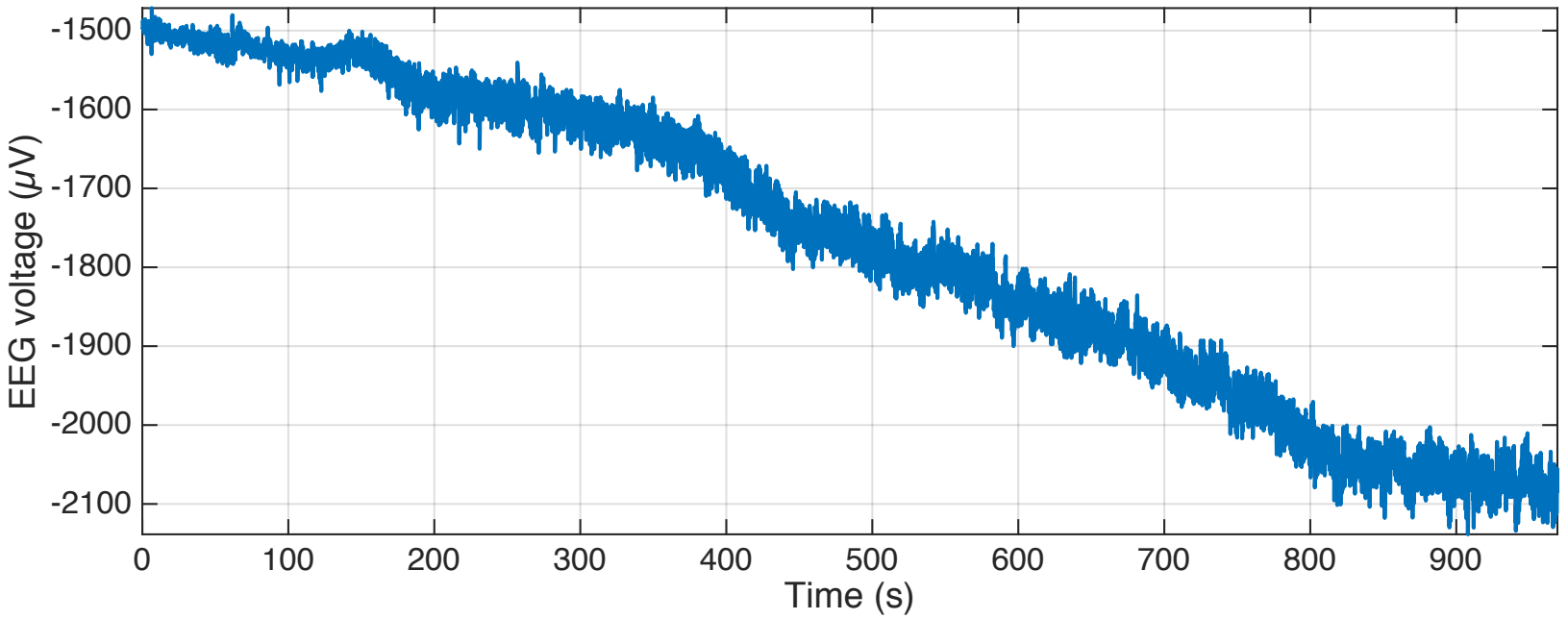
- Smooth splines:

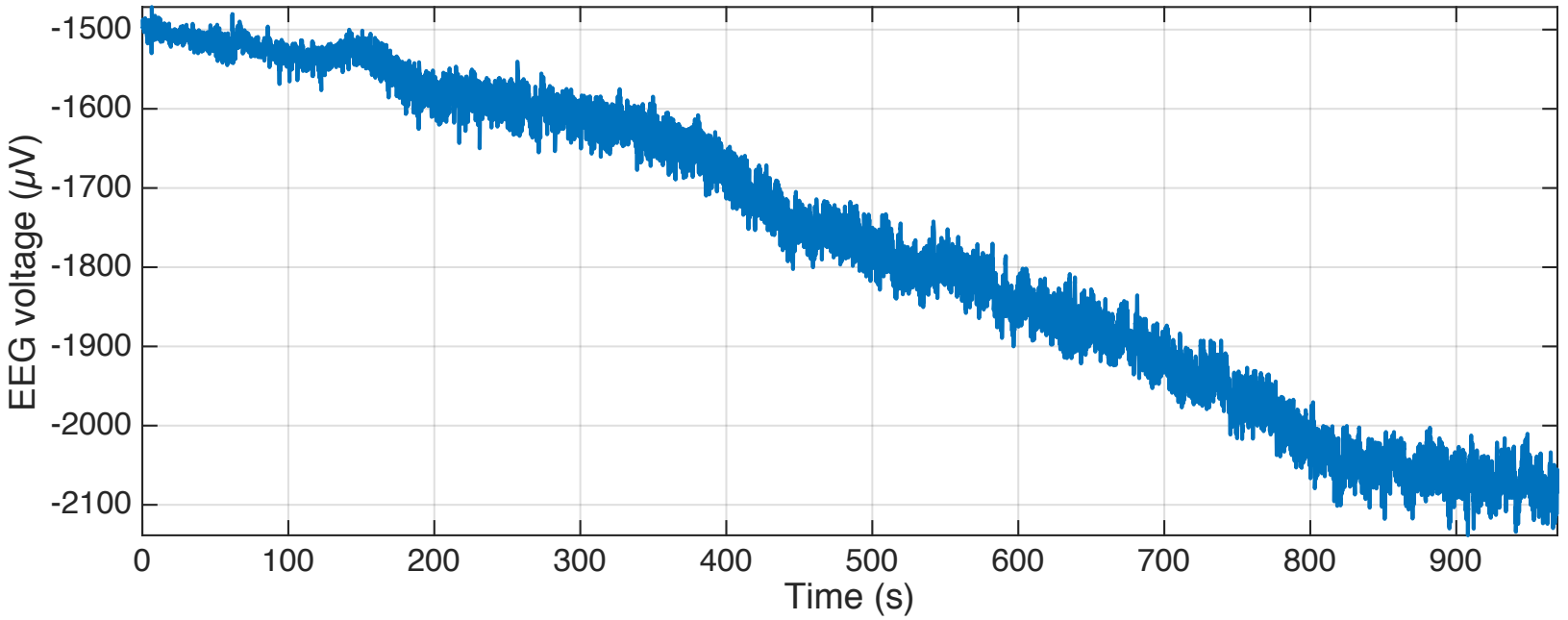
$$y = f(x, \beta_0, \beta_1, \ldots, \beta_K), \text{ with a constraint: } g(\beta) = 0$$

Recording EEG data

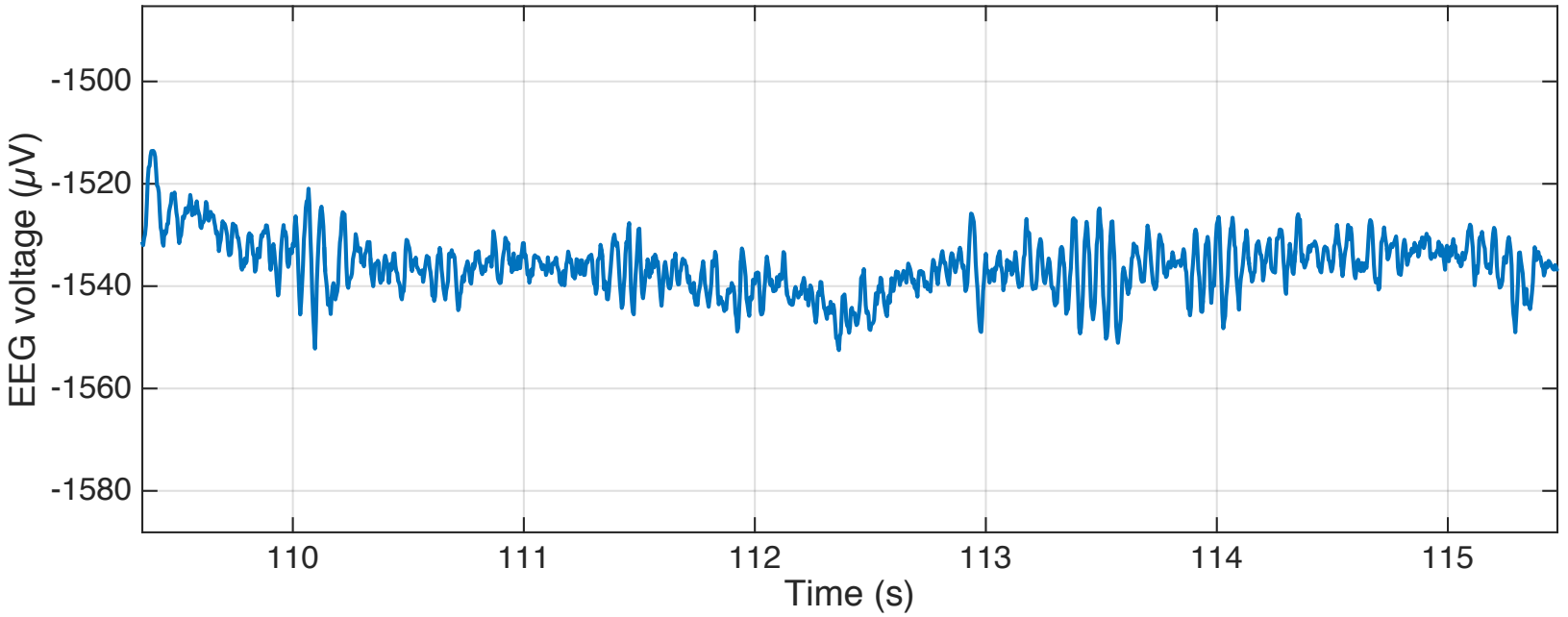
Example dataset: EEG recording during anesthesia

- Raw EEG data has many noise sources:
  - Drift due to sweat, electrode movement
  - Eyeblinks, movements
- First step in analysis is to remove the slow (smooth) “trends” in the data



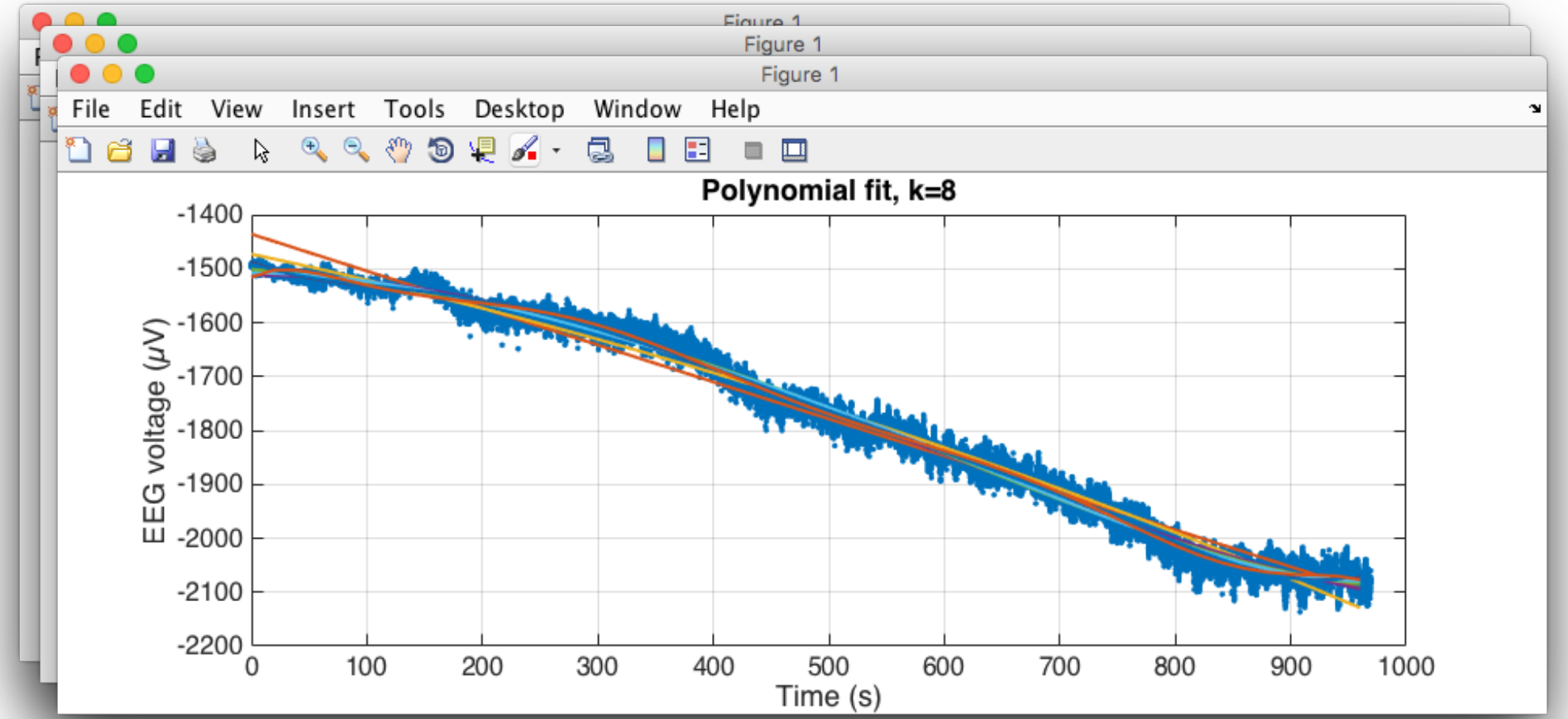

-1500
-1600
-1700
-1800
-1900
-2000
-2100
EEG voltage (μV)
0
100
200
300
400
500
600
700
800
900
Time (s)

- Zoom in:


-1500
-1520
-1540
-1560
-1580
EEG voltage (μV)
110
111
112
113
114
115
Time (s)

Method 1: Polynomial regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i,$$



Method 2: Splines

- A spline is a *piecewise polynomial* function

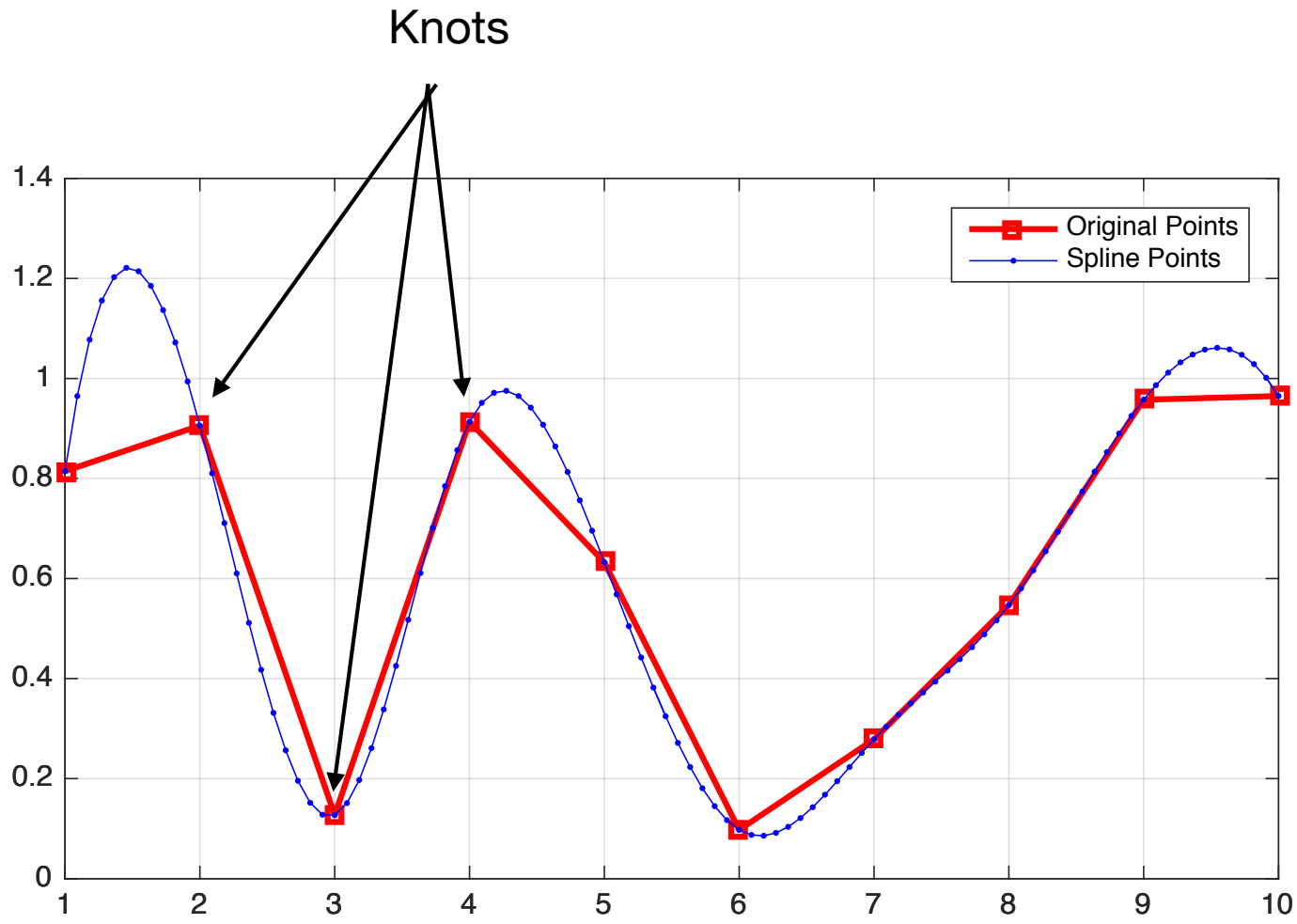
- Example: piecewise constant: piecewise linear:

$$y = \beta_0^{(1)} \text{ for } x \in [-\infty, K^{(1)})$$
$$y = \beta_0^{(2)} \text{ for } x \in [K^{(1)}, K^{(2)})$$
$$y = \beta_0^{(3)} \text{ for } x \in [K^{(2)}, K^{(3)})$$
$$\ldots$$
$$y = \beta_0^{(M)} \text{ for } x \in [K^{(M-1)}, \infty]$$

$$y = \beta_0^{(1)} + \beta_1^{(1)} x \text{ for } x \in [-\infty, K^{(1)})$$
$$y = \beta_0^{(2)} + \beta_1^{(2)} x \text{ for } x \in [K^{(1)}, K^{(2)})$$
$$y = \beta_0^{(3)} + \beta_1^{(3)} x \text{ for } x \in [K^{(2)}, K^{(3)})$$
$$\ldots$$
$$y = \beta_0^{(M)} + \beta_1^{(M)} x \text{ for } x \in [K^{(M-1)}, \infty]$$

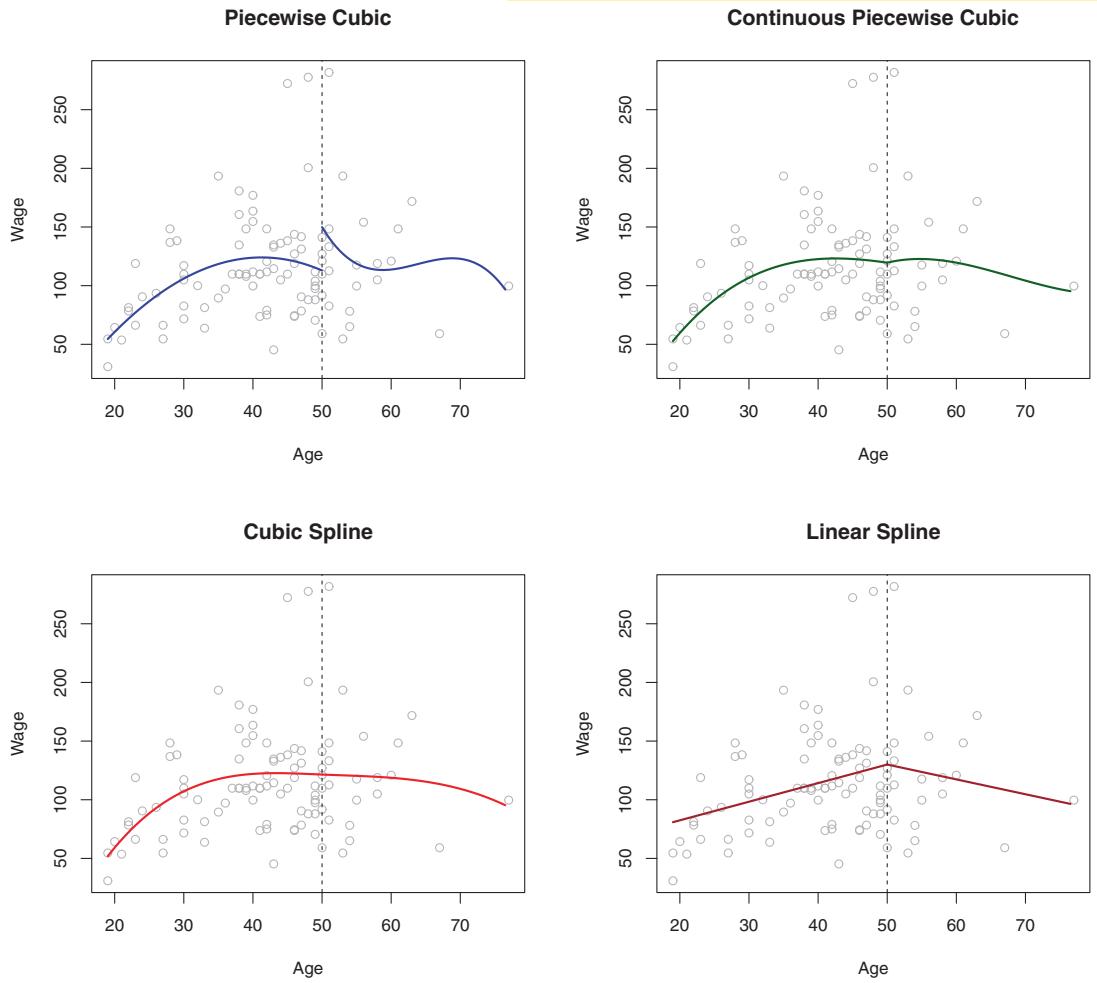
Knots: {-Infinity, K1, K2, …, KM}

Spline interpolation



Splines

- A spline is constrained to be continuous at the knots
- It is also required to have continuous derivatives



Comparing splines and polynomial fits

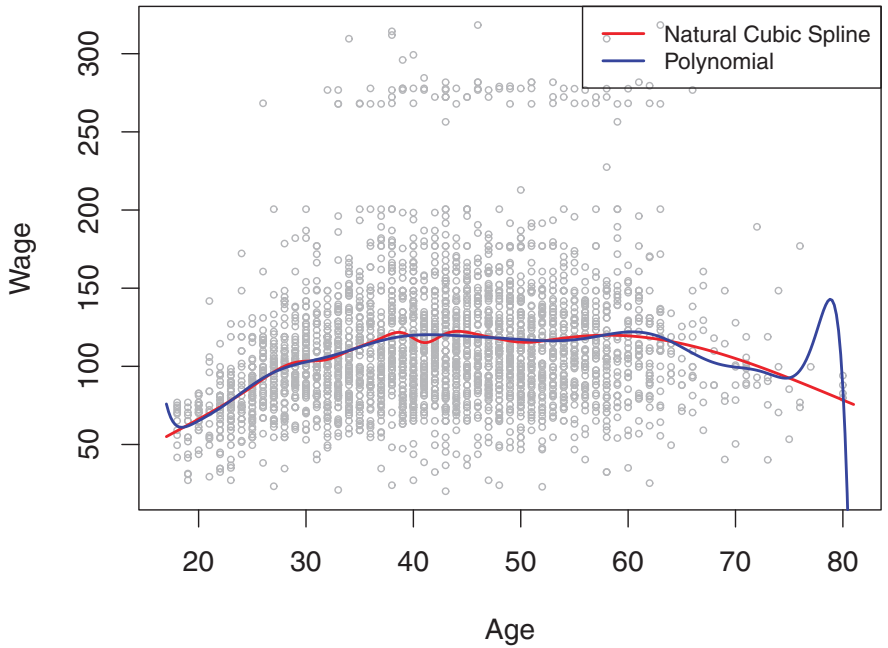
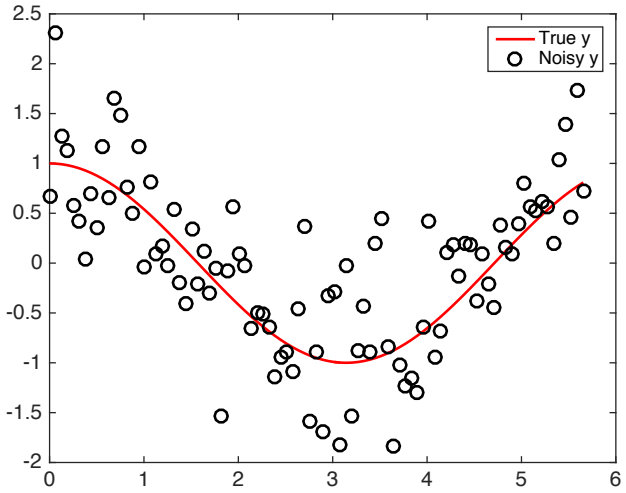
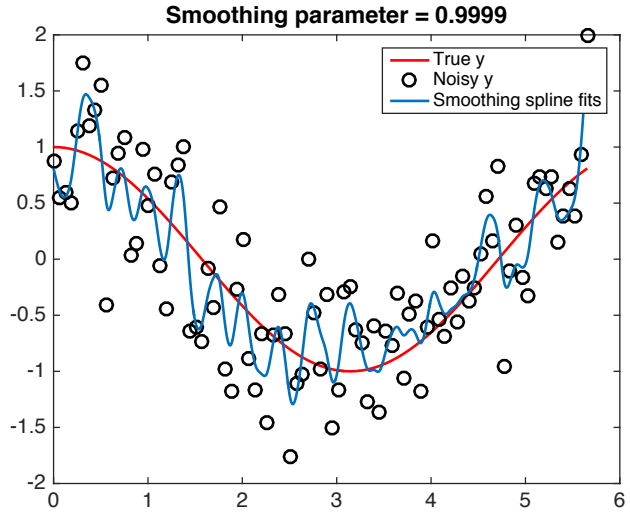


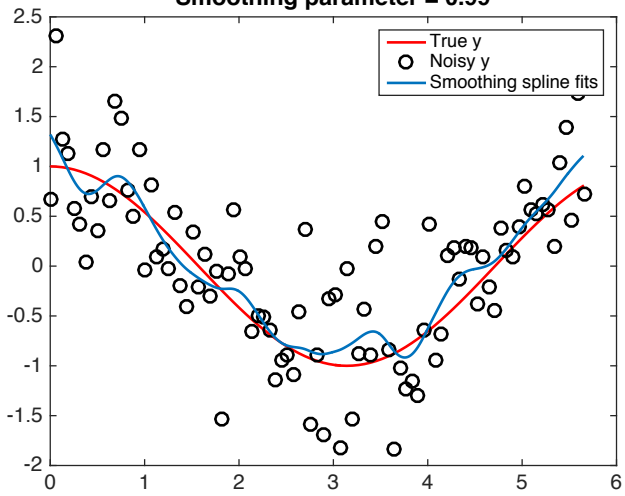
FIGURE 7.7. *On the* `Wage` *data set, a natural cubic spline with 15 degrees of freedom is compared to a degree-15 polynomial. Polynomials can show wild behavior, especially near the tails.*

Cubic smoothing splines

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

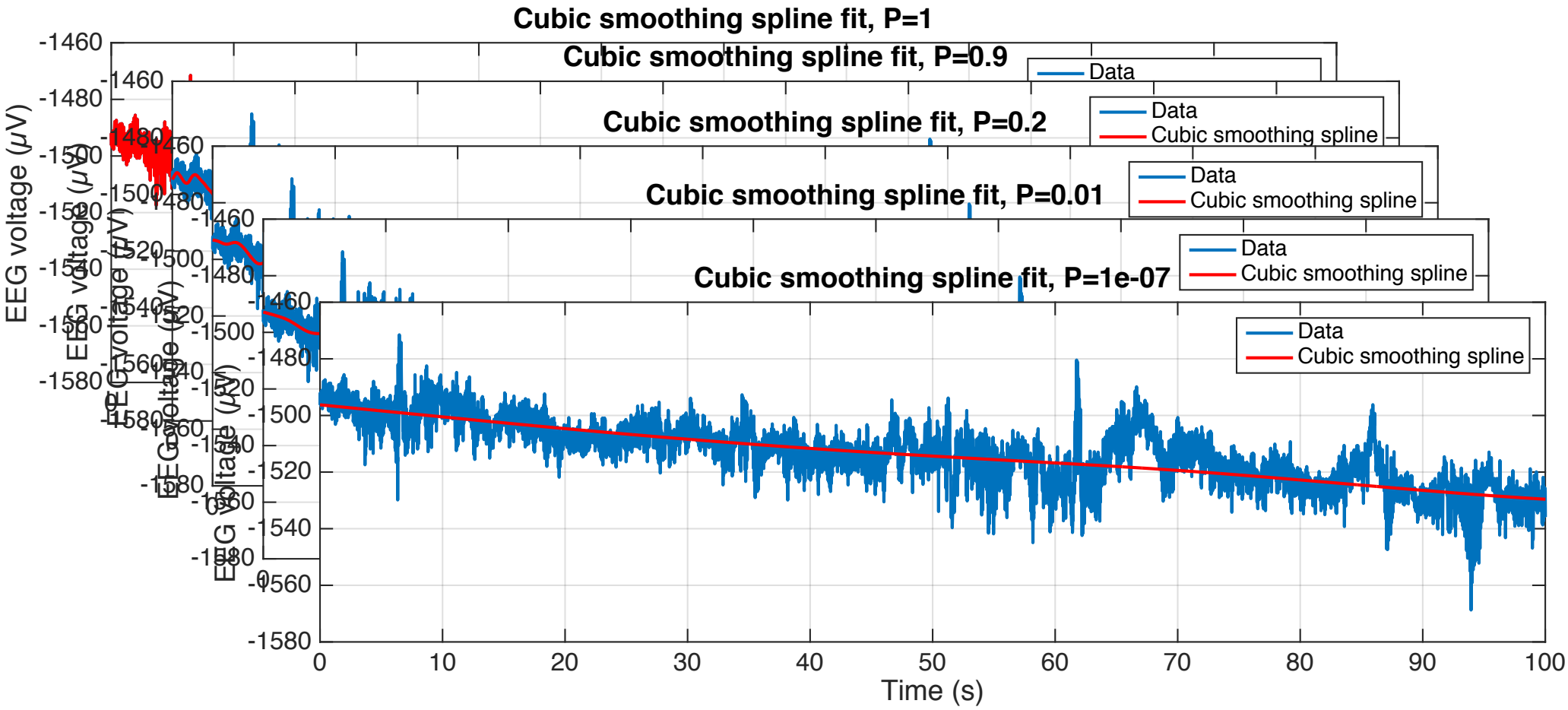






Cubic smoothing splines

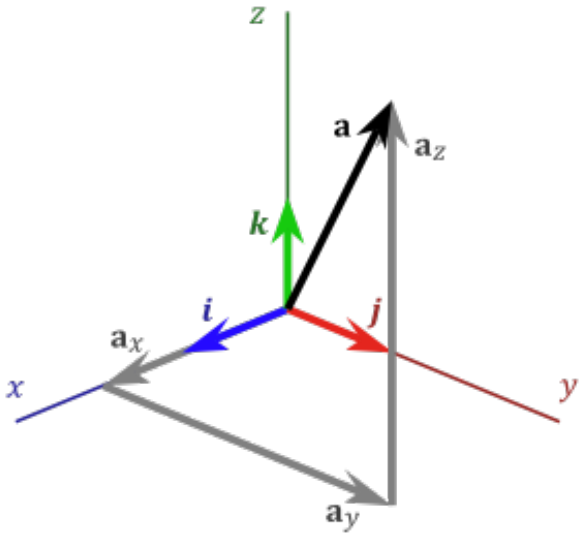
$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$



Basis functions

- Recall that a basis for a vector space is a set of vectors which can be combined to form any other vector:

$$\vec{v} = a_1 \vec{e_1} + a_2 \vec{e_2} + a_3 \vec{e_3} = \sum_{i=1}^{D} a_i \vec{e_i}$$



- The number of basis vectors is the dimension of the space

- Any (smooth) function can be written as a sum of (an infinite number of) polynomials (Taylor expansion). The polynomials are a basis for the space of functions.

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots$$

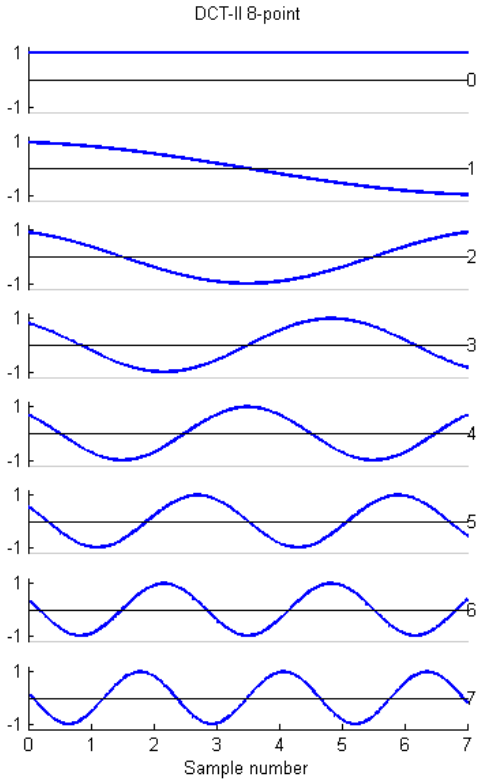
$$f(x) = \sum_{i=0}^{\infty} \beta_i x^i$$

Other basis functions

- Polynomials
- Piecewise polynomials (splines)
- Step functions
- Sin and cos functions (Fourier series)

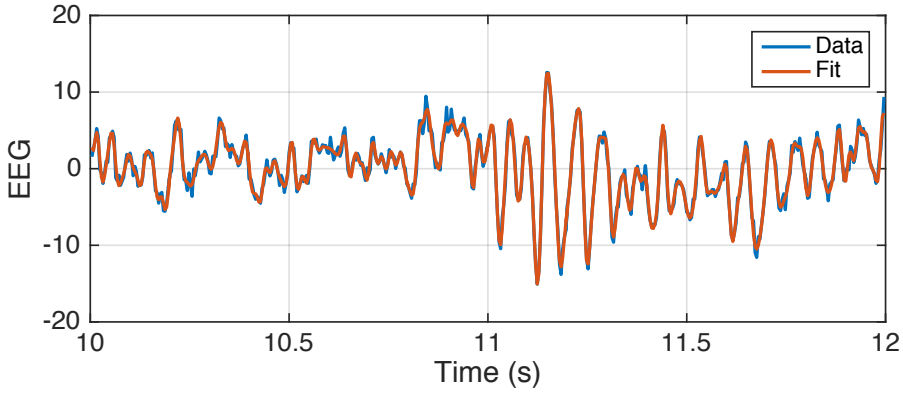
$$
\begin{aligned}
C_0(X) &= I(X < c_1), \\
C_1(X) &= I(c_1 \le X < c_2), \\
C_2(X) &= I(c_2 \le X < c_3), \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \le X < c_K), \\
C_K(X) &= I(c_K \le X),
\end{aligned}
$$

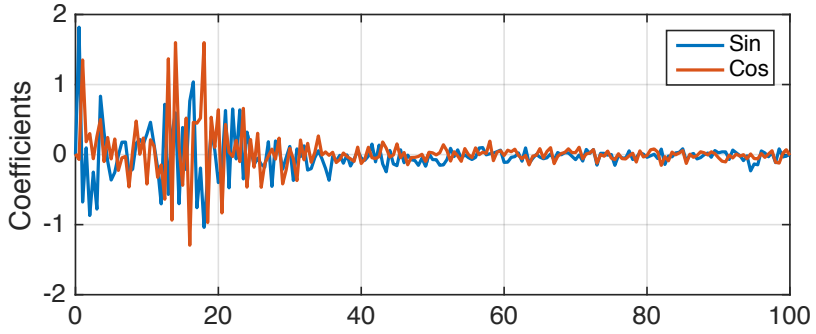
$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \ldots + \beta_K C_K(x_i) + \epsilon_i.$$

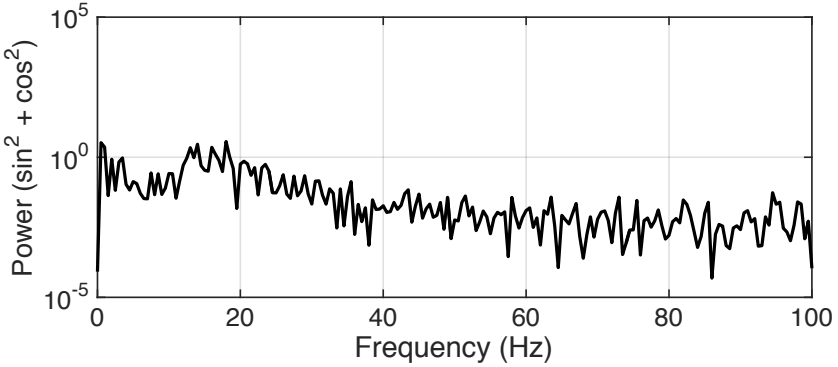


Sin and Cos functions as basis for regression

```
%% Fit sin and cos waves

frequencies = [0:1/2:50];
s = []; c = [];
for j=1:length(frequencies)
    f = frequencies(j);
    s(:,j) = sin(2*pi*f*T);
    c(:,j) = cos(2*pi*f*T);
end

X = [s,c];

%% Linear regression

b = fitlm(X, data, 'intercept',false);
yhat = b.predict;
```

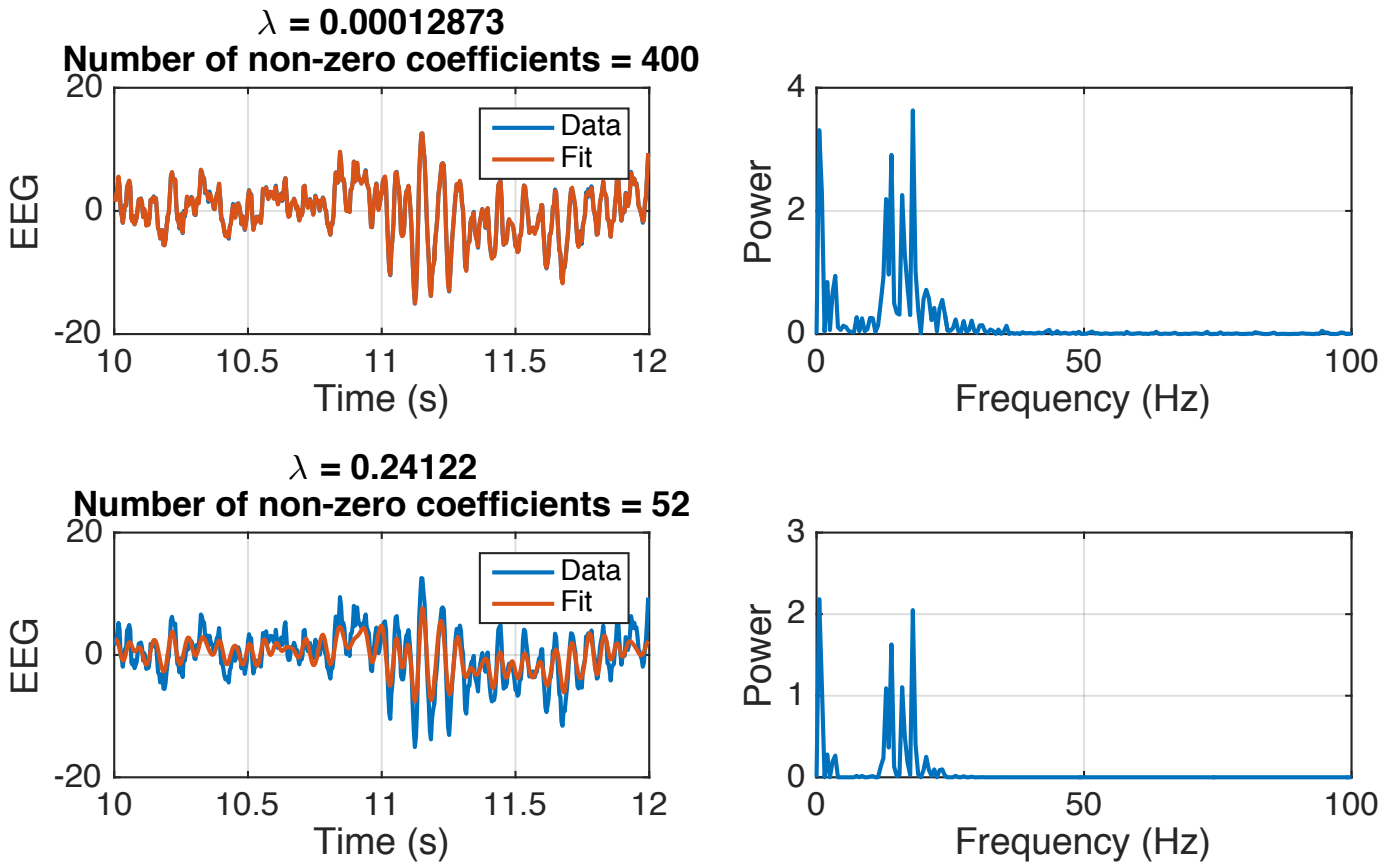






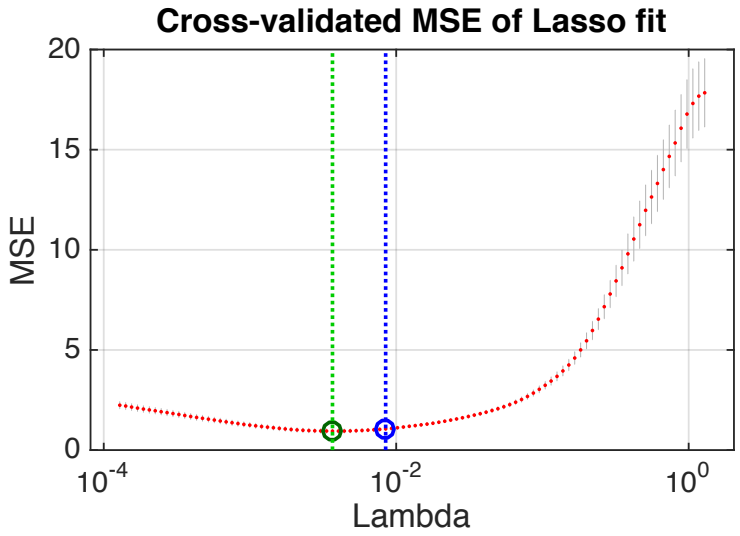
Using LASSO to regularize the fit

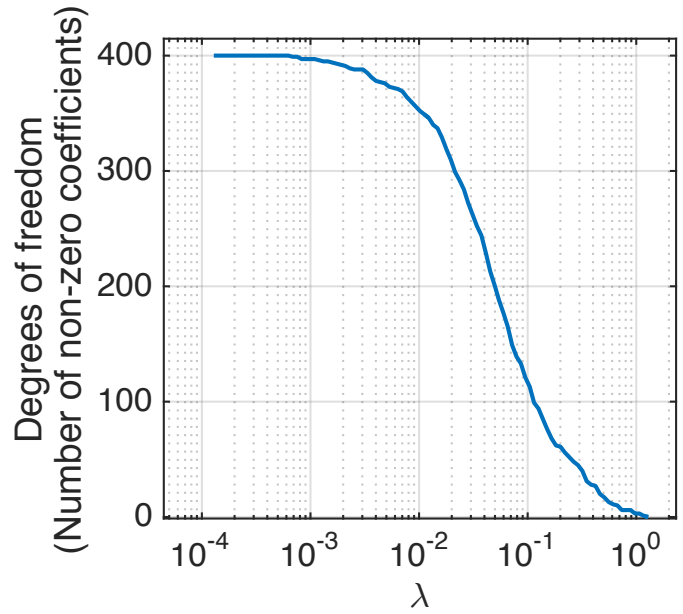
```
[b,fitinfo] = lasso(X, data);
```



Let MATLAB do the cross-validation for us

```
[b,fitinfo] = lasso(X, data,'CV',10);
lassoPlot(b,fitinfo,'PlotType','CV')
```





Degrees of freedom for cubic splines

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

- Each cubic polynomial has 4 parameters (degrees of freedom)
- However, the constraints at the knots reduce the number of degrees of freedom:
    - Function must be continuous
    - Function must have continuous first derivative
    - Function must have continuous second derivative
- With these constraints, each new knot adds only 1 additional degree of freedom

The spline basis representation

- Any cubic spline can be written as a weighted sum of "spline basis" functions

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$h(x,\xi) = (x-\xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise,} \end{cases}$$

- Each additional knot adds one additional parameter (beta)

$$h(x = \xi, \xi) = ?$$

- Question 1: What are the values of h and its first and second derivatives *at the knots themselves?*

$$\frac{d}{dx} h(x,\xi)\big|_{x=\xi} = ?$$

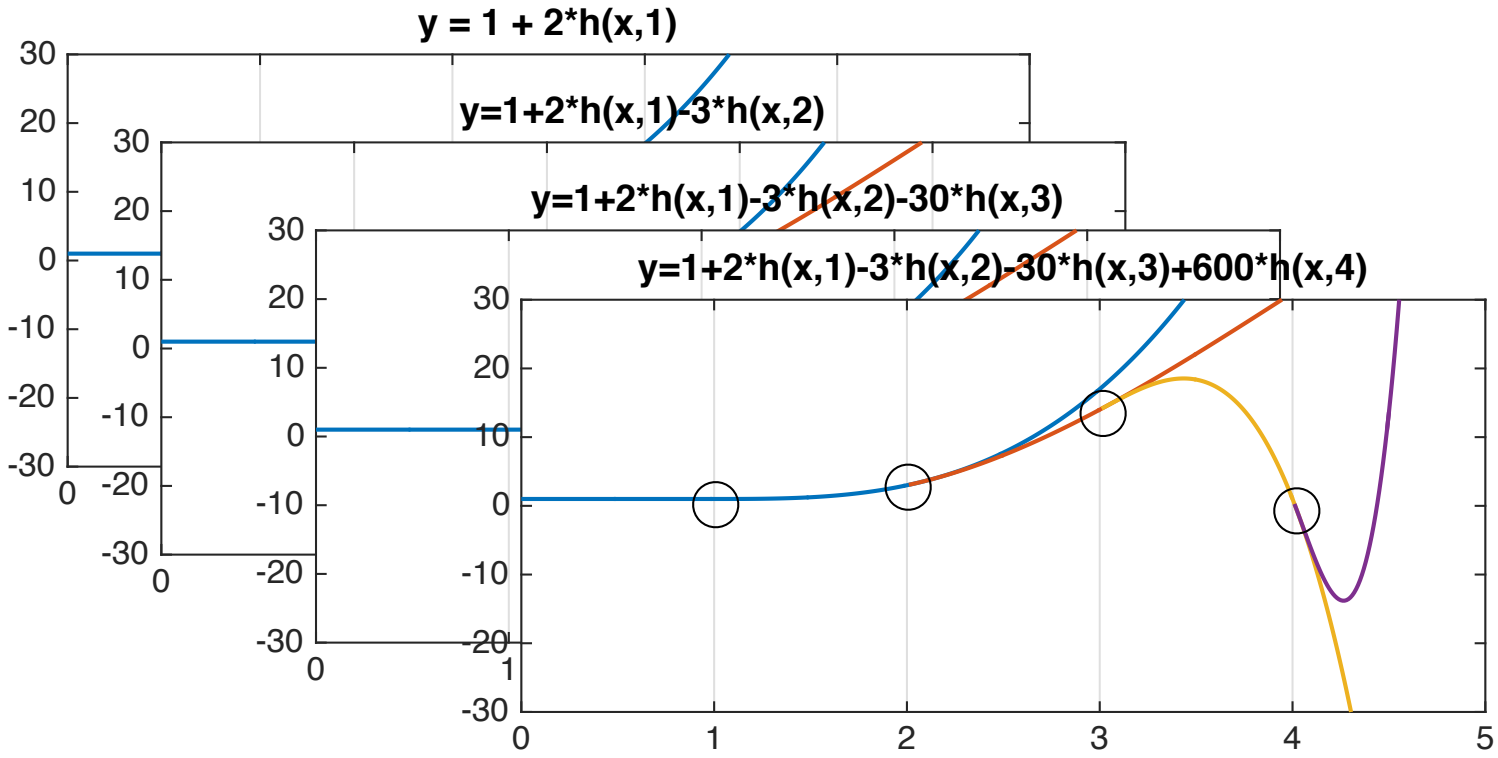
$$\frac{d^2}{dx^2} h(x,\xi)\big|_{x=\xi} = ?$$

- Question 2: What does this imply about the smoothness of the cubic spline function?

The spline basis representation

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$h(x,\xi) = (x-\xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise,} \end{cases}$$

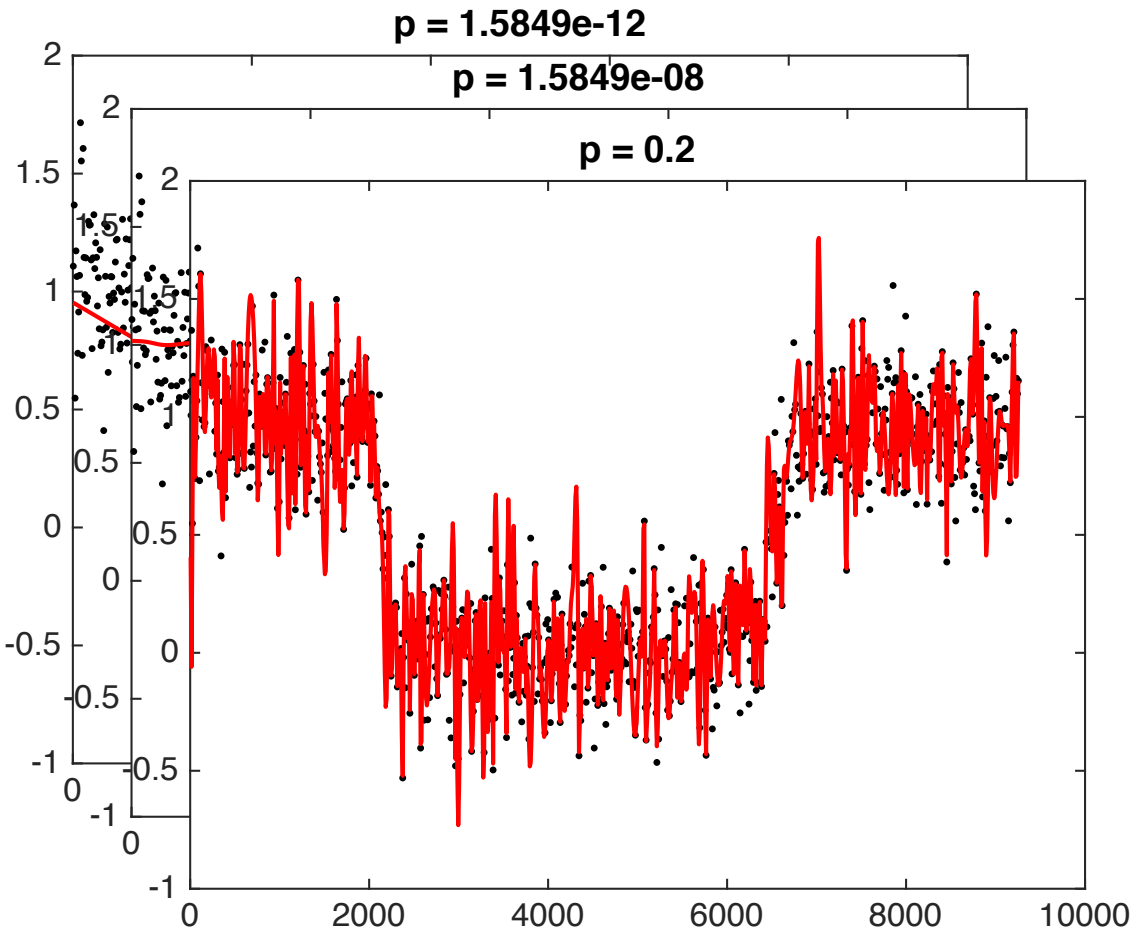


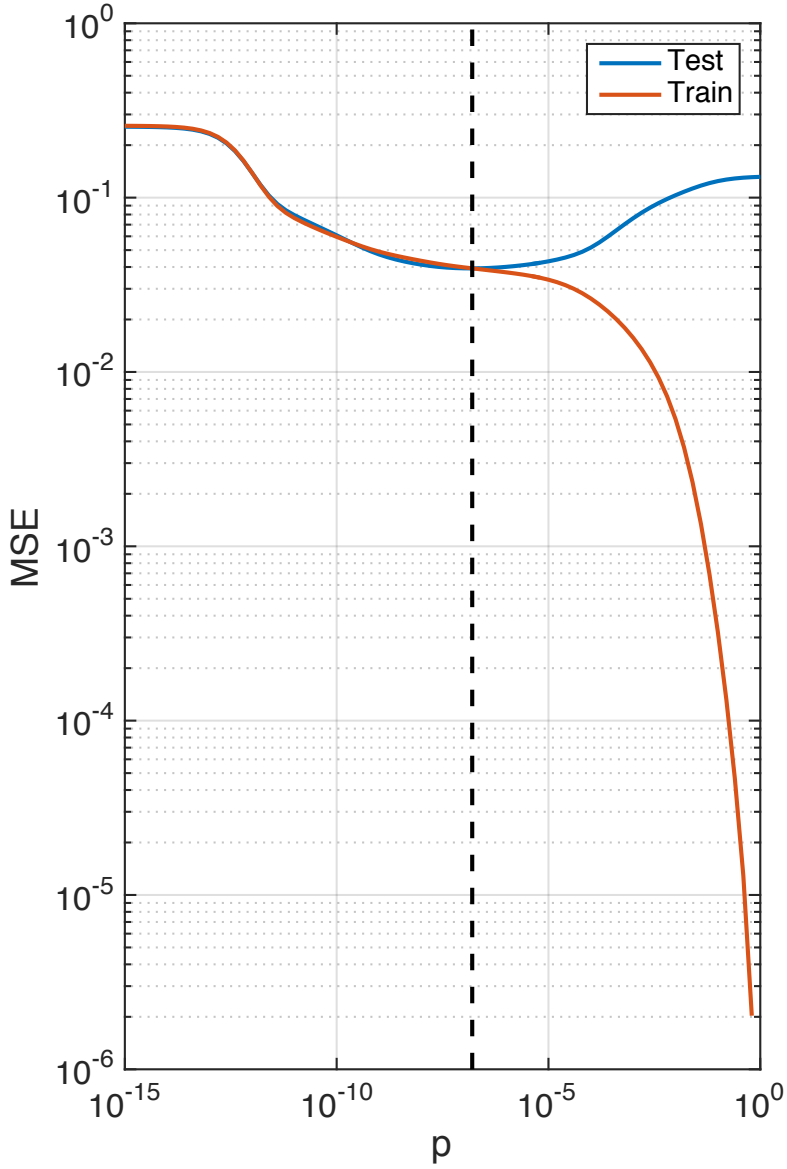
Smoothing splines

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

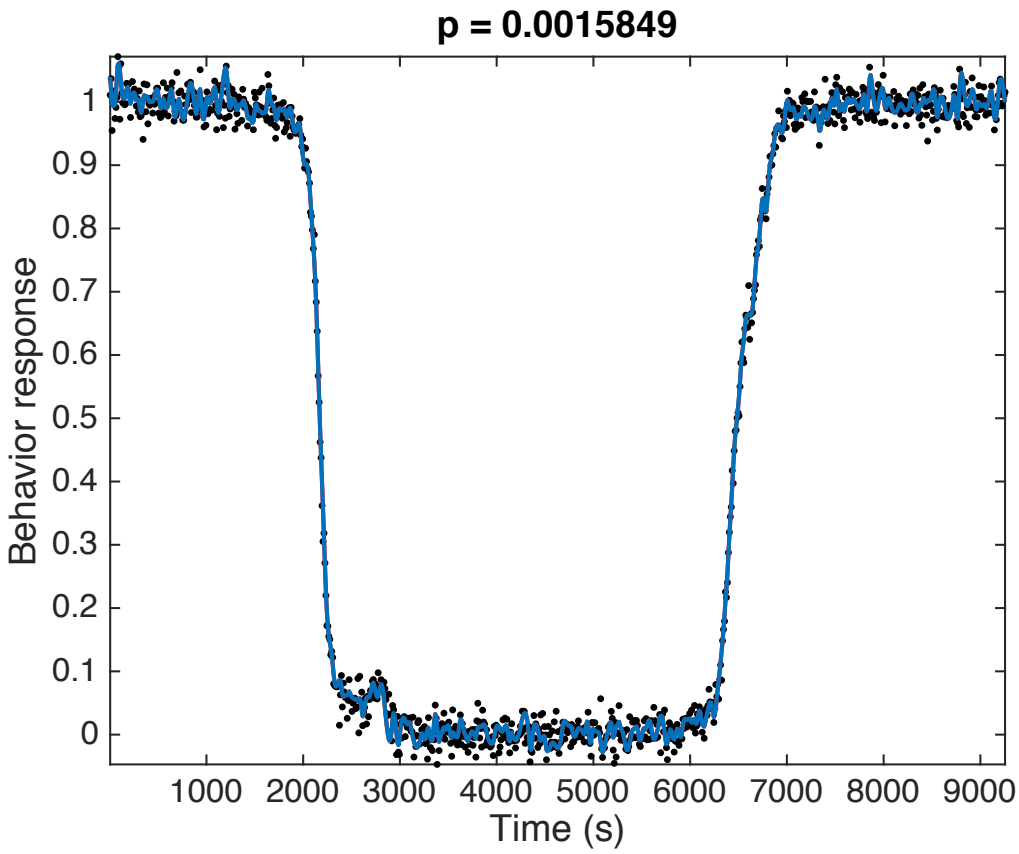
- The smoothing spline that minimizes the cost function has knots at the data points themselves
- It is constrained to have continuous first *and* second derivatives at each of the knots
- How many *effective degrees of freedom* (df_{lambda}) does the spline fit have?
    - If lambda = 0, df_{lambda} = n (the number of data points)
    - If lambda = Infinity, df_{lambda} = 2 (slope & intercept of straight line)
    - In between, df_{lambda} varies continuously from n to 2

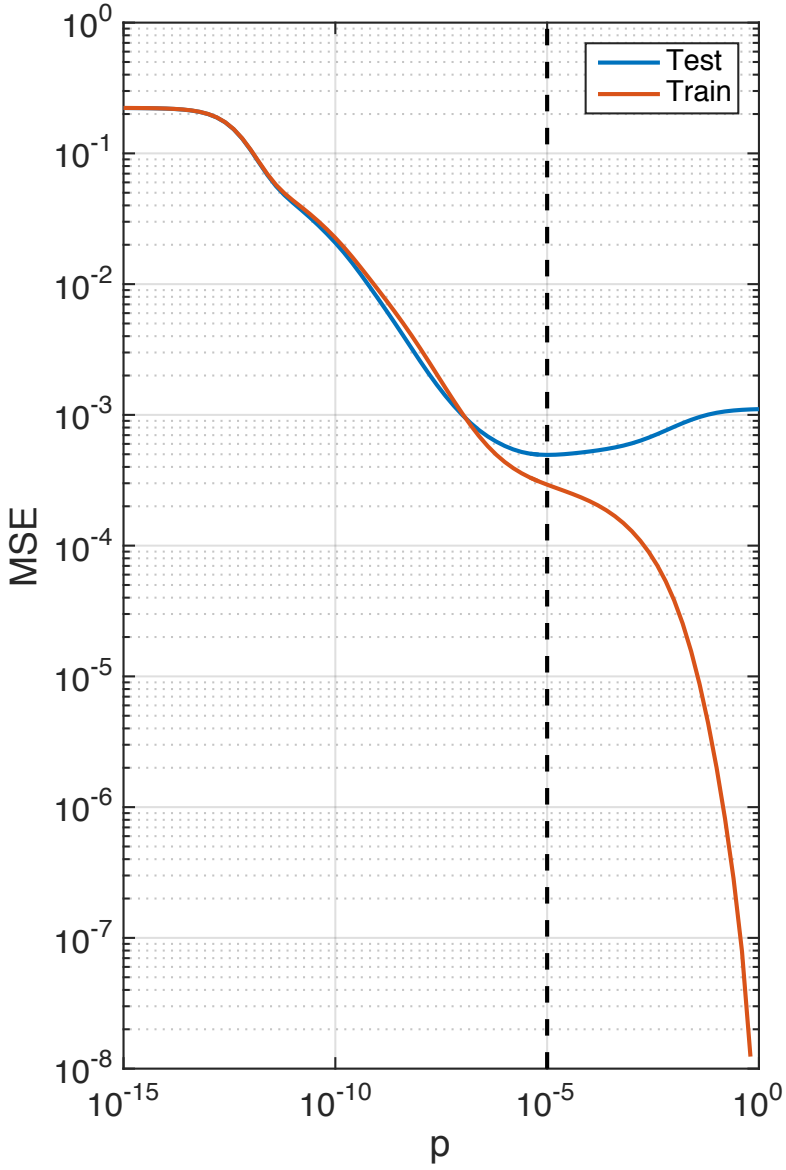
Smoothing spline:
Optimal smoothness depends on noise





Smoothing spline:
Optimal smoothness depends on noise





Recap of nonlinear regression methods

- Polynomial regression
- Basis functions
- Regression splines
  - Use piecewise polynomials (linear, cubic, …)
  - Knots: constrain the function to be continuous
- Smoothing splines
  - Regularization enforces smoothness
  - Knots are located at the data points