5.5 Multidimensional splines

Multidimensional splines

Each of the approaches discussed in this weeks course material has multidimensional analogues.

Let now assume that $X \in \boldsymbol{R}^2$ and we have a basis functions $h_{1k}(X_1)$, $k = 1, \ldots, M_1$ for representing functions of coordinate X_1, and likewise a set of M_2 functions $h_{2k}(X_2)$ for X_2.

Then the $M_1 \times M_2$ dimensional tensor product basis is defined by

$$g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2), \quad j = 1, \ldots, M_1, k = 1, \ldots, M_2$$

and it can be rewritten as a two-dimensional function:

$$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk} g_{jk}(X).$$

One dimensional smoothing splines generalize to higher dimensions, too. For x_i, y_i with $x_i \in \boldsymbol{R}^d$ we seek a d-dimensional regression function $f(x)$. The idea is to set up the problem

$$\min_f \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda J[f],$$

where J is an appropriate penalty functional for stabilizing a function f in $\boldsymbol{R}^d$.

For example, in $\boldsymbol{R}^2$ we have

$$J[f] = \int \int_{R^2} \left[\left(\frac{\partial^2 f(x)}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(x)}{\partial x_2^2} \right)^2 \right] dx_1 dx_2.$$

Optimizing the above criterion leads to a smooth two-dimensional surface, known as **thin-plate spline**. The solution here has the form

$$f(x) = \beta_0 + \beta^\top x + \sum_{j=1}^{N} \alpha_j h_j(x),$$

where $h_j(x) = \|x - x_j\|^2 \log \|x - x_j\|$. Here h_j is an example of **radial basis functions**. The coefficients α_j are found by plugging this solution back into the optimization criterion.

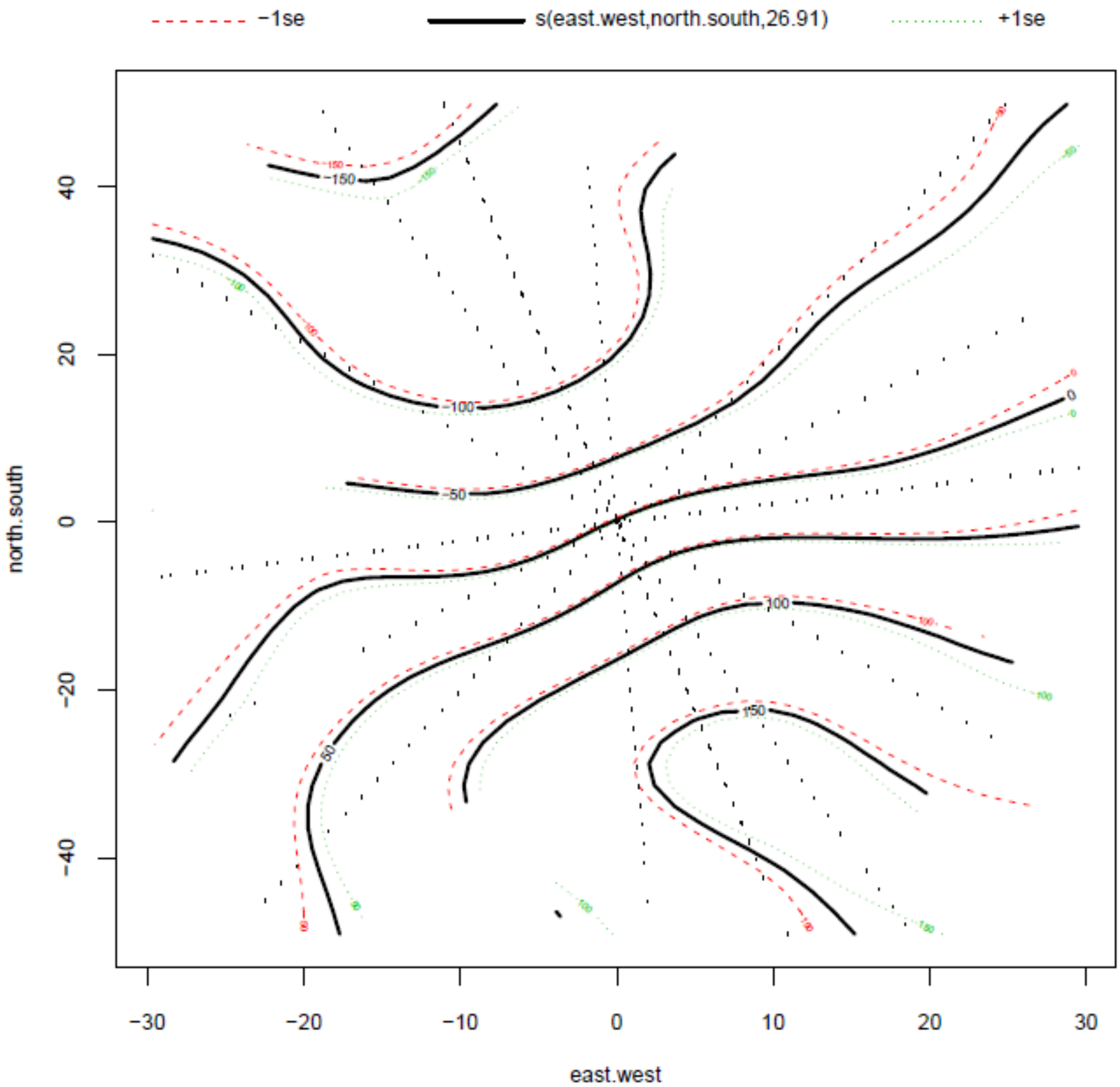
Example: Fitting a thin plate spline in R

A thin-plate spline fit is implemented in the package **mgcv**:

```
require(mgcv)

galaxy <-read.delim("/course/data/galaxy.data", sep=",")

fit=gam(formula = velocity~s(east.west,north.south),data = galaxy)
plot(fit)
```



Fits in higher dimensions may be calculated as well, but are not visualised as easily. Alternatively, a loess smooth can be fitted:

```
galaxy <-read.delim("/course/data/galaxy.data", sep=",")
require(mgcv)
```

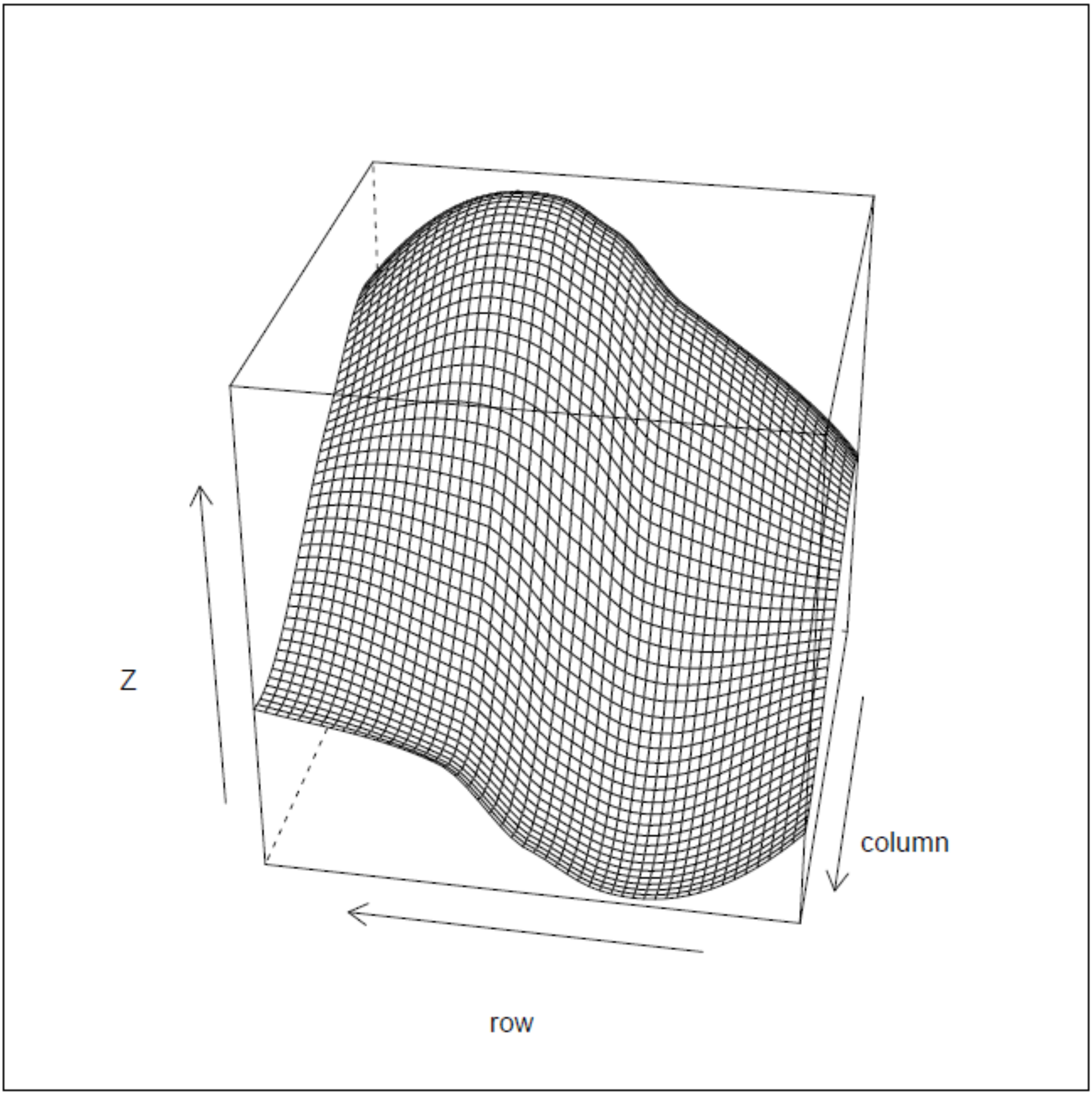
```
loc.lin.reg = loess(formula = velocity ~ north.south +
                      east.west, data = galaxy, span = 0.7, degree = 2,
                    normalize = T, family = "gaussian")
attach(galaxy)

xgrid = seq(min(north.south), max(north.south), length.out = 50)
ygrid = seq(min(east.west), max(east.west), length.out = 50)

new.coords = matrix(data = NA, nrow = 2500, ncol = 2)
new.coords[, 1] = as.vector(outer(rep(1, 50), xgrid))
new.coords[, 2] = as.vector(outer(ygrid, rep(1, 50)))

new.coords = as.data.frame(new.coords)
names(new.coords) <- c("north.south", "east.west")

Z = predict(loc.lin.reg, newdata = new.coords)
Z = matrix(data = Z, ncol = 50)
library(lattice)
wireframe(Z, screen = list(z = 170, x = -60))
```


Z
column
row

Activity in R: 3D visualisation

Consider the **`galaxy`** dataset:

```
galaxy <- read.csv("./data/galaxy.data")
head(galaxy,n = 2)
```

```
  row.names east.west north.south angle radial.position velocity
1         3  8.462789   -38.17317 102.5            39.1     1769
2         4  7.964978   -35.92769 102.5            36.8     1749
```

Generate a plot of the three-dimensional point cloud **`east.west`**, **`north.south`**, **`velocity`** using the $cloud()$ function in the **`lattice`** library in R.