Splines and bases

- Before using splines for spatial statistics, we'll introduce splines for nonparametric regression in 1-d.
- Splines are piecewise functions connected at locations called knots. The resulting overall function can be used to represent smooth relationships between covariates/locations and outcomes.
- Various types of splines can be used as bases.
- A basis is a set of simple functions that can be added together in a weighted fashion to form more complicated functions.
- A simple basis is the polynomial basis. Consider a polynomial regression model:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i.$$

- The basis is the set of power functions from which one can construct the regression function: $\{1, x, x^2, x^3, x^4, x^5, x^6, \ldots\}$

- In the regression model,

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

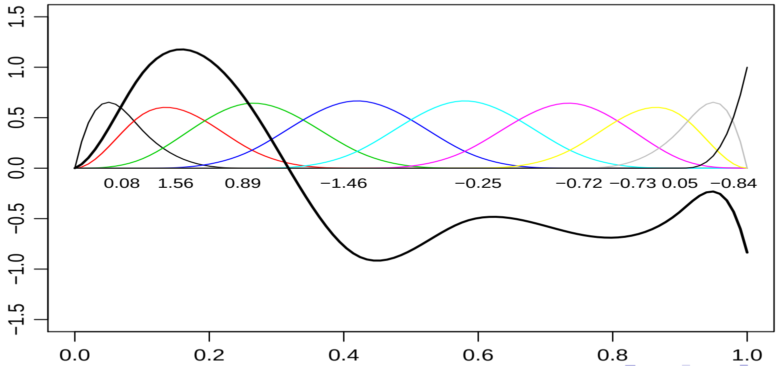
  the function is represented using four basis functions: $f_i = \sum_{j=0}^{3} b_j(x_i)\beta_j$, where $b_j(x) = x^j$, and the weights are the coefficients, β_j.
- We can express the function for all the observations jointly as $f = B\beta$, where the matrix B contains the basis functions, $b_j(x_i)$, evaluated for each of the observations, e.g. for 4 coefficients:

$$\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} 1 & b_1(x_1) & b_2(x_1) & b_3(x_1) \\ 1 & b_1(x_2) & b_2(x_2) & b_3(x_2) \\ 1 & \vdots & \vdots & \vdots \\ 1 & b_1(x_n) & b_2(x_n) & b_3(x_n) \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}$$

- The polynomial basis has some problems. In particular, it is unstable at the boundaries and can swing wildly, because the estimate of each coefficient is influenced by all the data.
- Instead we'll consider three types of spline models: regression splines, penalized splines, and penalized regression splines.

Regression splines

- Regression splines fit smooth regression relationships using spline bases, of which there are a number of different types. One example with nice statistical and numerical properties is the b-spline basis, which we can again represent as: $f_i = \sum_{j=1}^{J} b_j(x)\beta_j$. An example with six knots (9 basis functions) is given here:



Regression splines

- Regression splines fit smooth regression relationships using spline bases, of which there are a number of different types. One example with nice statistical and numerical properties is the b-spline basis, which we can again represent as: $f_i = \sum_{j=1}^{J} b_j(x)\beta_j$.
- The number of basis functions in this case is the number of knots plus three.
- As with polynomial regression, the key is how many spline functions (how many knots) to use and where to put the knots. Too many knots lead to undersmoothing and too few to oversmoothing.

Smoothing splines

- Smoothing splines solve the knot number and placement problem by putting a knot at every data point and penalizing the wiggliness of the function. The smoothing spline is a solution to minimizing the objective function:

  $$\sum_i (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx,$$

  where λ is a penalty parameter than controls how much to penalize wiggly functions (functions with second derivatives that are large in magnitude).

- The solution is a tradeoff between goodness of fit (the sum of squares term) and wiggliness of the smooth function (the second term).
- The key is deciding what the value of λ should be to determine the trade-off. Again, various forms of cross-validation are popular.
- Smoothing splines have some computational problems in that they take a long time to compute when working with large datasets because there are as many basis functions as there are data points.

Penalized (regression) splines

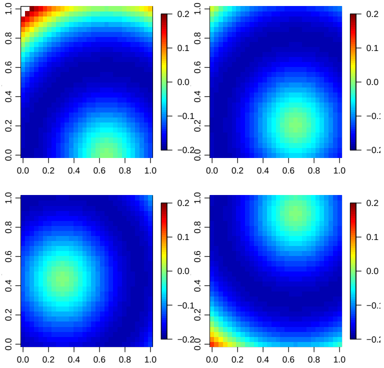
- A compromise between regression splines and smoothing splines is to have enough basis functions/knots to allow for a more wiggly function that you expect and then to penalize the wiggliness as in smoothing splines.
- The result is a flexible model, computationally-feasible, that avoids some issues with knot placement and the number of knots.
- The optimization problem is to minimize

  $$\sum_i (y_i - B_i^T \beta)^2 + \lambda \beta^T S \beta$$

  where the matrix, S, is constructed using the spline basis chosen, and B is the basis matrix, as previously. Here the penalty makes sure the coefficients don't get too big, so the function can't swing too wildly.

Splines in 2-d

- Thin-plate splines are smoothing splines in 2-d, and are the result of minimizing $\sum(z_i - g(s_1, s_2))^2 + \int\int \left(\frac{\partial^2 g}{\partial s_1^2}\right)^2 + 2(\frac{\partial^2 g}{\partial s_1 \partial s_2})^2 + \left(\frac{\partial^2 g}{\partial s_2^2}\right)^2 ds_1 ds_2$, which penalizes wiggliness in both directions and at an angle.
- Here are some thin-plate spline basis functions:



Thin plate splines in practice

`Optional, but please check out` **`mgcv`** `package in R.`

- Simon Wood has created a very computationally-efficient version of thin-plate splines for use in two-dimensional smoothing, including spatial smoothing, and this is implemented in gam() in the mgcv library (which also fits more general additive regression models).
- Wood's version is a penalized thin plate spline that uses matrix manipulations to reduce the effective number of basis functions used for the thin-plate spline ($k < n$). The idea is to choose k large enough to be sufficiently flexible and small enough (in large datasets) to be computationally feasible.
- The penalty is chosen using either a form of cross-validation (generalized cross-validation (GCV), a computationally-efficient version of cross-validation), by REML or by another method.

- Wood's algorithm starts with n knots and then does some computational tricks to get down to k, so if n is very large, it can choke.
- For very big problems, Wood also allows one to choose a set of knots such that $k < n_{knots} < n$ for even more efficiency.
- An example in gam() is:
  - `model=gam(z~s(x,y,bs='tp',k=50)+ s(v1) + s(v2))`
- Make sure k is big enough! (Might need to change defaults.)
- The output of gam() reports 'edf', the effective degrees of freedom used in fitting the data, which should be $1 \leq edf \leq k$.