`Interpolations.jl` - mathematical background

Tomas Lycken
tomas.lycken@gmail.com
github.com/tlycken

December 26, 2014

Abstract

This document intends to outline the mathematical foundations of the `Interpolations.jl` package, in order to make it easier to understand how the library works, as well as to change or extend its behavior. Since it shows off the derivations that underlie the code, it is also intended to help figuring out if a bug stems from coding or mathematical mistakes.

Contents

1 Introduction

Interpolations.jl uses *B-splines*[1], or *basis splines*, to interpolate a data set with a specified degree. In short, the idea is to use a set of basis functions with limited support, and in each interval between two data points, construct a linear combination of the basis functions which have support in that interval, each yielding a piece in a piecewise defined function with support across the entire data set. B-splines are defined to have the maximum degree of continuity, with minimal support. Specific details on the general mathematics of such interpolations have been extensively presented by Thevenaz, Blu, and Unser [1].

Before going into specific implementation details, we must define a few core concepts.

1.1 Interpolation degree

The interpolation degree decides what continuity properties the interpolant (i.e. the object which represents the interpolated data set) has; for example, a linear interpolation is a piecewise linear function, which is continuous but has discontinuous derivatives (or gradients, in higher dimensions).

Because of how B-splines are defined, the interpolation degree also decides the *support* of the *spline functions*. For a linear interpolation, the function value is decided by the value at *two* data points, so its support is two[2].

1.2 Grid cells

When looking at a discrete grid, one can consider the data points to define either cell boundaries (with one grid cell spanning the domain between two data points) or cell centers (with one grid cell spanning the domain closest to a single data point). On uniform, 1-spaced grids, this corresponds to having cell boundaries either at integers $1, 2, \ldots$ (where the data points are specified), or at half-integers $\frac{1}{2}, \frac{3}{2}, \ldots$ (exactly centered between data points).

Each interpolation type defines a *grid behavior*, specified by `OnGrid` or `OnCell`. However, interpolations of even (odd) degree will use an odd (even) number of points, and we require them to be symmetric. Thus, even (odd) degree interpolations will use `OnCell` (`OnGrid`) behavior inside the domain, regardless of the `OnCell` or `OnGrid` specification in the interpolation type; `OnCell`/`OnGrid` will be used solely to determine where the edges of the domain lie.

[1] `http://en.wikipedia.org/wiki/B-spline#Definition`

[2] For this reason, linear B-splines are often referred to as *2nd order*, which may be a source of confusion since the interpolating function itself is linear, i.e. of first order. In `Interpolations.jl`, we will try to avoid this confusion by referring to interpolation degree by "linear", "quadratic" etc.

1.3 Boundary conditions

For higher interpolation degrees (specifically, from quadratic interpolation and up), the support of the B-splines is too large for the interpolating scheme to be able to figure out the values near the edges of the data sets. In order to close the equation systems at the edges, boundary conditions are used.

1.4 Extrapolation behavior

Somewhat orthogonal[3] to the concepts outlined above, is the concept of *extrapolation*, i.e. evaluation of the interpolant outside the domain defined by the data set. For some types of extrapolation, this behavior is defined by a translation of the interpolation coordinate to somewhere inside the domain (e.g. periodic or reflecting boundaries), while for other types it entails a separate calculation entirely (e.g. linear extrapolation).

2 Quadratic B-splines

2.1 Cell definition

A quadratic polynomial requires three coefficients, and thus quadratic B-splines are *three-point*, i.e. the interpolating spline function in one grid cell takes three provided data points into account. It is natural to center the grid cells around the middle point, so that the spline function around x_i is is defined on the interval $\delta x \in [-\frac{1}{2}, \frac{1}{2}]$, where $\delta x = x - x_i$ and i ranges from 1 to the number of data points N.

2.2 General governing equations

Assuming a symmetric spline basis, one finds that

$$y_i(\delta x) = c_{i-1}\left(\tfrac{1}{2} - \delta x\right)^2 + c_i\left(\tfrac{3}{4} - \delta x^2\right) + c_{i+1}\left(\tfrac{1}{2} + \delta x\right)^2 \qquad (1)$$

Furthermore, we require that on cell boundaries, the function values and first derivatives are continuous, which yields $y_i(\frac{1}{2}) = y_{i+1}(-\frac{1}{2})$ and $y_i'(\frac{1}{2}) = y_{i+1}'(-\frac{1}{2})$. Inserting these in each-other, we find the following equation for each data point:

$$\frac{1}{8}c_{i-1} + \frac{3}{4}c_i + \frac{1}{8}c_{i+1} = v_i$$

where v_i is the provided data point at x_i. Considering the corresponding equation for all provided data points simultanesously, we form a tridiagonal equation system that we need to solve for the coefficients c_i. However,

[3] Although the implementational separation of extrapolation behavior is computationally sound, it does allow for some interesting, yet probably nonsensical, combinations of interpolation degrees, boundary conditions and extrapolation behavior. In cases where the extrapolation behavior is defined as constant or reflecting, it will make sense to specify matching boundary conditions, but other combinations are entirely supported by `Interpolations.jl`. No guarantees are left that the results make sense, though...

since we have two more unknowns than we have data points (namely c_0 and c_{N+1}), we need to apply boundary conditions to close the system.

2.3 Boundary conditions

The boundary conditions are implemented by specifying further conditions on the function value and/or its derivatives at the outer boundary (or, in some cases, close to it). For simplicity, we will here only describe the lower boundary close to $x = 1$, yielding equations for c_0 - the same arguments are applied at the upper end of the interval, specifying c_{N+1}.

We will have use for the following expressions for the first and second derivatives of the interpolating function:

$$y_i'(\delta x) = \frac{1}{2}\left[c_{i-1}(2\delta x - 1) - c_i \cdot 4\delta x + c_{i+1}(2\delta x + 1)\right]$$
$$y_i''(\delta x) = c_{i-1} - 2c_i + c_{i+1}$$

which were obtained simply by derivation of equation (1) with respect to δx.

2.3.1 Flat boundary conditions

`Flat` boundary conditions are implemented by imposing that $y' = 0$ at the outermost cell boundary.

2.3.1.1 `Quadratic{Flat,OnGrid}`

In this case, the edge is at $x = 1$, corresponding to $(i, \delta x) = (1, 0)$. Inserting this in $y' = 0$ yields

$$y'(x_{\text{edge}}) = y_1'(0) = \frac{1}{2}\left[-c_0 + c_2\right] = 0$$
$$\Rightarrow -c_0 + c_2 = 0$$

Thus, the first few rows of the equation system become

$$\begin{bmatrix} -1 & 0 & 1 & & & \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & & & \\ & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & & \\ & & \ddots & \ddots & \ddots & \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ \vdots \end{bmatrix}$$

2.3.1.2 `Quadratic{Flat,OnCell}`

$$y'(x_{\text{edge}}) = y_1'\left(-\frac{1}{2}\right) = \frac{1}{2}\left[-2c_0 + 2c_1\right] = 0$$
$$\Rightarrow -c_0 + c_1 = 0$$

$$\begin{bmatrix} -1 & 1 & & & \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & & \\ & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & \\ & & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ \vdots \end{bmatrix}$$

2.3.2 Linear, or natural, boundary conditions

For linear boundary conditions (in many contexts referred to as "natural"), we want the second-order derivative at the boundary to equal zero, i.e. $y''(x_{\text{edge}}) = 0$.

2.3.2.1 `Quadratic{Linear,OnGrid}`

$$y''(x_{\text{edge}}) = y_1''(0) = c_0 - 2c_1 + c_2 \Rightarrow c_0 = 2c_1 - c_2$$

$$\frac{1}{8}c_0 + \frac{3}{4}c_1 + \frac{1}{8}c_2 =$$
$$\left(\frac{2}{8} + \frac{3}{4}\right)c_1 + \left(\frac{1}{8} - \frac{1}{8}\right) = c_1 = v_1$$

2.3.2.2 `Quadratic{Linear,OnCell}`

$$y''(x_{\text{edge}}) = y_1''\left(-\tfrac{1}{2}\right) = c_0 - 2c_1 + c_2 \Rightarrow c_0 = 2c_1 - c_2$$

$$\frac{1}{8}c_0 + \frac{3}{4}c_1 + \frac{1}{8}c_2 =$$
$$\left(\frac{2}{8} + \frac{3}{4}\right)c_1 + \left(\frac{1}{8} - \frac{1}{8}\right) = c_1 = v_1$$

References

[1] P. Thevenaz, T. Blu, and M. Unser. "Interpolation revisited [medical images application]". In: *Medical Imaging, IEEE Transactions on* 19.7 (July 2000), pp. 739–758. ISSN: 0278-0062. DOI: 10.1109/42.875199. URL: http://bigwww.epfl.ch/publications/thevenaz0002.pdf.