MUS 470/470L Homework 3

Fall 2019

Cubic Spline Interpolation of Audio Signals

Due date: Thursday, November 14.

Write a program which takes a short wav file and interpolates the audio data with cubic splines. Use the follow input parameters:

1. s_R = the sample rate of the input file
2. s_I = the number of samples per interval for each cubic spline
3. k = the number of subintervals inside each interval
4. d = the degree of the cubic splines (handle $d = 1, 2, 3$)

Output should be a wav file of the same length as input, with values determined from the spline functions constructed to interpolate the original data.

Determination of intervals for splines

A default interval size can be $s_I = 1024$ samples. Another approach is to model a sound with assumed frequency, say 440 Hz. This would suggest a convenient interval size for each spline approximation to be one cycle of the signal, which is about 100 samples per cycle if we use $s_R = 44100$.

The next step is to use k subintervals for the spline functions on each interval. The number of subintervals is less than the number of samples per interval. For a data reduction which is comparable to MP3 compression, we can set the default number of intervals to be one tenth of the number of samples, so $k = s_I/10$.

Bases for splines

Two bases are commonly used: the truncated power basis, and the B-spline basis. Each can be constructed for any sequence of k sub-intervals, say $[u_0, u_1, \ldots, u_k]$, and a degree d for the polynomial pieces on each interval, and an order of continuity r. For simplicity, assume the intervals are of uniform width, say $w = u_i - u_{i-1}$. Instead of one order of continuity r, a vector of continuity conditions can be specified instead, such as $\mathbf{r} = (r_1, r_2, \ldots, r_{k-1})$ so that the spline functions agree to order of continuity r_i at the break-point u_i, which excludes the endpoints. The default order of continuity will be $r = d-1$, which is the maximum finite r for piecewise polynomials of degree d. To achieve $r = d$ immediately elevates r to infinity, which means the piecewise polynomial is just a single polynomial across all the intervals.

The vector space $V = P^k_{d,r}[u_0, u_1, \ldots, u_k]$ of all polynomial splines of degree d with order of continuity $r = d - 1$ on the intervals $[u_0, u_1, \ldots, u_k]$ has dimension $n = d + k$. So both of the bases will have n functions in them.

The truncated power basis functions are given by first choosing a knot sequence

$$\mathbf{t} = \{t_0, t_1, \ldots, t_{n-1}\}.$$

The simplest default should be to use the sequence u_i with d numbers appended to the left end of the sequence, so that for $d = 3$, for example, we get $t_0 = u_0 - 3w$, $t_1 = u_0 - 2w$, $t_2 = u_0 - w$, $t_3 = u_0$. Then the sequence continues

with $t_{3+i} = u_i$, up to $i = k-1$. So the final value is $t_{n-1} = u_{k-1}$. Also, $n = k + d - 1 = k + 2$ for $d = 3$. With this simple default we also get that $r = d - 1 = 2$.

The truncated power basis functions attached to the knot sequence are obtained by wrapping the knot values by the form $(t-c)_+^{d-1}$ to give the set:

$$\{(t-t_0)_+^{d-1}, (t-t_1)_+^{d-1}, \ldots, (t-t_{n-1})_+^{d-1}\}.$$

The B-spline basis for the same vector space V described above is obtained from another knot sequence, similar to $\mathbf{t}$ above, but this time appending d values on the right as well as the left of the sequence. The simplest case is to use

$$\mathbf{t} = \{t_0, t_1, \ldots, t_N\},$$

with same starting values $t_0, t_1, \ldots, t_{n-1}$, but continuing with $t_{3+k} = u_k$, and $t_{3+i} = u_k + (i-k)w$ for $i = k+1$ up to $i = k + d - 1$. So the final value is now $t_N = u_{k-1} + (d+1)w$.

A more general form for the knot sequence that gives a B-spline basis can be described by:

$$\{\alpha_0, \ldots, \alpha_d, u_1, \ldots, u_{k-1}, \beta_0, \ldots, \beta_d\},$$

where $\alpha_i \leq u_0$ and $\beta_i \geq u_k$, for $i = 0, \ldots, d$. (Check that the example cases described above adhere to these requirements.)

The B-splines associated to the knot sequence $\mathbf{t}$ above can be labeled

$$\mathcal{B}_0(t), \mathcal{B}_1(t), \ldots, \mathcal{B}_{N-d-1}(t).$$

Solving for an explicit form of any B-spline can be tedious, but it turns out that we can find the coefficients for a spline in terms of these B-splines and then evaluate a linear combination with those coefficients, without ever writing down any explicit forms. This is done through the use of the DeBoor algorithm for B-spline evaluation. In general it is not necessary for the knot sequence to have uniform separation w between values. The knot sequence can also have consecutive equal values, as long as the multiplicity does not exceed $d+1$. In the case of multiple values, the basis splines are chosen with consecutively lower degrees, which accomodates representation of functions with lower orders of continuity at the break points.

Solving for an interpolating spline

Let $\{f_0, \ldots, f_{n-1}\}$ be a basis of splines of one of the two types above, for a vector space of splines on a sequence of intervals as above. Now suppose we want to solve for a function f as a linear combination of these basis functions which passes through a given set of data points. If there are n data points (x_i, y_i), for $i = 0, \ldots, n-1$, then it may be possible to solve for such an f. In fact, as long as we have $t_i < x_i < t_{i+d+1}$, for each input x_i, where t_j refers to the B-spline knot sequence $\mathbf{t}$, then it will be possible to solve for f with a linear system approach. Let f be defined as:

$$f(t) = a_0 f_0(t) + a_1 f_1(t) + \cdots + a_{n-1} f_{n-1}(t).$$

Then we form each row of the linear system by plugging in each of the inputs x_i and setting equal to the output value y_i. Each row of the linear system then has the form:

$$a_0 f_0(x_i) + a_1 f_1(x_i) + \cdots + a_{n-1} f_{n-1}(x_i) = y_i.$$

Extracting the constants $f_j(x_i)$ as coefficients of the a_j, we can convert this to a row of an augmented matrix:

$$[f_0(x_i) \;\; f_1(x_i) \;\; \cdots \;\; f_{n-1}(x_i) \;\; | \;\; y_i].$$

The entire augmented matrix looks like this:

$$\left(\begin{array}{cccc|c} f_0(x_0) & f_1(x_0) & \cdots & f_{n-1}(x_0) & y_0 \\ f_0(x_1) & f_1(x_1) & \cdots & f_{n-1}(x_1) & y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_0(x_{n-1}) & f_1(x_{n-1}) & \cdots & f_{n-1}(x_{n-1}) & y_{n-1} \end{array}\right).$$

For example, if we set $d = 3$, then we can use as inputs x_i the set of endpoints and breakpoints of intervals as a starting point. But this is only $k + 1$ values, and we have $n = k + d = k + 3$. So we can insert two more values into this list. It is convenient to assign those values as $x_1 = u_0 + \frac{1}{2}w$ and $x_{n-2} = u_k - \frac{1}{2}w$. The rest of the sequence then uses the u_j, so that $\{x_0, x_2, \ldots, x_{n-3}, x_{n-1}\} = \{u_0, \ldots, u_k\}$. This sequence then satisfies the requirement for a unique solution, which is $t_i < x_i < t_{i+d+1}$.

Signal Modeling with Splines

Models of an audio signal with splines can be compared in various ways. An obvious way is to listen to the audio file and check for distortion. Another way is to look at the spectrum to see what has been significantly changed.

Once a set of B-spline coefficients is chosen for a model of an audio signal, it can become a starting point for further modeling and reduction of data. One way to reduce the amount of data stored is further interpolate the B-spline coefficients with another set of B-splines.

An example of this method is *cycle interpolation*. In this method, the chunk of data which determines the interval length is a cycle. This is not well-defined, but can be approximated for instance in the case of a short sample of an instrument playing a predetermined pitch such as A 440. Since one cycle has length about 1/440 of a second, the audio file can be scanned for zero crossings, taking the zero crossing closest to the expected length. This way each interval represents one cycle, and is approximated with a set of $k + d$ B-spline coefficients. Since neighboring cycles are quite similar, one can experiment with interpolation of cycle coefficients.