Spline Library in DROP

v3.14 4 October 2017

Overview

Framework Symbology and Terminology

1. Predictor Ordinates: The segment independent/input values.
2. Response Values: The segment dependent/output values.
3. C^0, C^1, and C^2 Continuity: C^0 refers to base function continuity. C^1 refers to the continuity in the first derivative, and C^2 refers to continuity in the second.
4. Local Piece-wise Parameterized Splines: Here the space formulation is in the local variate space that spans 0 to 1 within the given segment – this is also referred to as piece-wise parameterization.
5. Bias: This is the first term in the Spline Objective Function – essentially measures the exactness of fit.
6. Variance: This refers to the second and the subsequent terms in the Spline Objective Function – essentially measures the curvature/roughness.

Motivation

1. Definition: "**Spline** is a sufficiently smooth polynomial function that is piecewise-defined, and possesses a high degree of smoothness at the places where the polynomial pieces connect (which are known as *knots*). [Spline (Wiki), Judd (1998), Chen (2009)]
2. Drivers:
    a. Lower degree, gets rid of oscillation associated with the higher degrees [Runge's phenomenon (Wiki)]
    b. Easy, accurate higher degree smoothness specification
3. Basic Spline: Covered in [Spline (Wiki), Bartels, Beatty, and Barsky (1987), Judd (1998), De Boor (2001), Fan and Yao (2005), Chen (2009), Katz (2011)].

4. History: Schoenberg (1946), Ferguson (1964), Epperson (1998).

Document Layout

References

- Bartels, Beatty, and Barsky (1987): *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*.
- Chen, W. (2009): *Feedback, Nonlinear, and Distributed Circuits* **CRC Press**.
- De Boor, C. (2001): *Practical Guide to Splines (Revised Edition)* **Springer**.
- Epperson (1998): History of Splines *NA Digest* **98 (26).**
- Fan, K., and Q. Yao (2005): *Non-linear Time Series: Parametric and Non-parametric Methods* **Springer**.
- Ferguson, J. (1964): Multi-variable curve interpolation *J ACM* **11 (2)** 221-228.
- Judd, K. (1998): *Numerical Methods in Economics* **MIT Press**.
- Katz, M. (2011): *Multi-variable Analysis: A Practical Guide for Clinicians and Public Health Researchers* **Cambridge University Press**.
- Schoenberg, I. (1946): Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions *Quart. Appl. Math.* **4** 45-99 112-141.
- Spline (Wiki): *Wikipedia Entry for Spline*.

Calibration Framework

Introduction

1. Definition: Calibration is the process of inferring the latent state elastic properties from the specified inputs.
    - Calibration takes both the "mandatory" and the "desirable" classes of inputs to fully determine the elastic properties.
    - It makes sense to generate the calibration micro-Jacobians right at the calibration time.
2. Classes of static fields: Elastic and inelastic
    - Elastic Fields => The actual Latent State response variables that will be calibrated to. More generally, any latent state response field (Markov or not) that is inferred will be an elastic parameter.
    - Inelastic Fields => The Latent State predictor ordinate variables – these are often explicitly "chosen" or "designed in".
        - Typically inelastic fields correspond to constitutive properties (e.g., dimensions of a solid body, instruments composing a curve, etc.)
        - Inelastic properties may also impose invariant, calibration independent edge/boundary behavioral constraint on the elastic ones.
        - Design Parameters (such as the C^k continuity parameters, roughness penalty derivative order, etc.) are inelastic parameters too, since they do not vary with changes to the calibrator input.
3. Calibrator Creation: On creation, objects acquire specific values for the constitutive inelastic fields. Volatile Latent State elastic fields may as yet be undefined.
    - Setting of the elastic fields => Latent State Elastic fields adjust or vary to the combination of inelastic fields + inputs (external), and are set by the calibration process.
    - Change of inputs => Change of external calibration inputs changes only those Latent State elastic properties, not the inelastic ones.

4. Calibration is Inference: Since calibrated parameters are used for eventual prediction, calibration is essentially inference. Bayesian classification (an alternate, generalized calibration exercise) is inference too.
    - The terms calibration/inference/estimation are all sometimes used analogously. Where estimation estimates parameters, it performs inference. Where it predicts, it performs prediction.
    - Infer the Past vs. Predict the Future => You may infer the past quantification metric, as well as predict the future quantification metric/manifest measure. Therefore, in that sense, inference/prediction is relative only to the current time (and using earlier/later information).
5. Calibration and entity-variate focus:
    - De-convolving the instrument entity/measure combination is necessary for the extraction of the parameter set.
    - Parameter calibration/parameterization etc. inherently involve parsimonization across the latent state predictor ordinate variate state space – this is where the models come in.
6. Latent State Construction off of hard/soft signals: Hard Signals are typically the truthness signals. Typically reduce to one calibration parameter per hard observation, and they include the following:
    - Actual observations => Weight independent true truthness signals
    - Weights => Potentially indicative of the truthness hard signal strength

    Soft signals are essentially signals extracted from inference schemes. Again, typically reduce to one calibration parameter per soft inference unit, and they include the following:
    - Smoothness signals => Continuity, first, second, and higher-order derivatives match – one parameter per match.
    - Bayesian update metrics => Inferred using Bayesian methodologies such as maximum likelihood estimates, variance minimization, and error minimization techniques.
7. Directionality Bias: Directionality “bias” is inherent in calibration (e.g., left to right, ordered sequence set, etc.) – this simplifies the solution space significantly, as it avoids simultaneous non-linearity. Therefore, the same directional bias also exists in the calibration nodal sequence.

8. Truthness/Smoothness vs. Information Propagation: In segment-by-segment calibration challenges associated with inferring a composite Latent State, if the inference is based purely off of the truthness measurements, the information directionality/propagation/flow is less relevant. Notions such as C^k are important primarily owing to the smoothness axioms.
    - In general, it is trivial to get the segment elastics to respond to (via inference) purely truthness signals. However, if the existence of additional directionality/propagation/flow criteria is posited, those need to be accommodated too (splines constructed through the C^k criteria are one common way).
9. Head/Tail Node Calibration: Calibration of the head (and sometimes the tail) node is typically inherently different from the other nodes, because the inputs needed/used by it could be different. The other nodes use continuity/smoothness parameters, which the head node does not.
10. Parameter Space Explosion: Generally not a problem as long as it is segment-localized (in linear systems parlance, as long the Latent State matrix is tri-diagonal, or close to it), i.e., local information discovery does not affect far away nodes/segments.
    - Also maybe able to use optimization techniques to trim them.
11. Live Calibrated Parameter Updating: Use automatic differentiation to:
    - Estimate parametric Jacobians (or sub-coefficient micro-Jacobians) to the observed product measures.
    - Re-adjust the shifts using the hard-signal strength.
    - Update the parameters from the calculated shifts.
    - Re-construct the curve periodically (for increments, as opposed to a full re-build).
12. Spline Segment Calibrator: For a given segment, its calibration depends only on the segment local value set – the other inputs come from the prior segments (except in the case of “left-most” segment, whose full set of inputs will have to be extraneously specified).
13. Block Diagonalization vs. Segment/Segment Calibration: Since segment-by-segment span calibration occurs through C^k transmission, it will effectively be block diagonal if it is a linear system, and hence computationally efficient (this also allows for explicitly setting segment level controls). In other words, although the local matrix is dense, the span-level matrix is still sparse,
14. Calibration Perspective of Supervised vs. Unsupervised:

- Supervised – Alternate View => Since supervised learning depends upon a training step, post supervised activity may be viewed as a post-inference systems, where the inference/calibration has already occurred during the training step (and parsimonization into the parameter step). So, post-training, the supervised machine is only used for prediction.
- Unsupervised – Alternate View => Here, inference and prediction happen simultaneously.
- Hybrid Supervised/Unsupervised => Clearly no TRUE unsupervised are possible (as there are prior views on linearity, Markov nature, error process, basis spline representation choice, etc.). So unsupervised systems are typically mostly hybrid systems.

Latent State Specification

1. Latent State Specification: The latent-state here refers to the state whose dependent response values we wish to calibrate/infer as a function of the predictor ordinate. For e.g., in fixed income finance, the Prevailing Interest Rate, the survival, the forward rate, FX spot/forward would each constitute a potentially separate latent state that needs to be inferred.
2. Latent State Quantification Metric: A given latent state may be described by one/more alternate, mutually exclusive quantification metric. Again, the discounting latent state may be quantified using a discount factor, zero rate etc.
3. Manifest Measure: The Latent State may be inferred using a variety of external experimental measurements each of which produces a convolved signal of the latent state. Each such signal is referred to as a manifest metric. Again, the discounting latent state may be inferred from the cash instrument manifest metric, the swap rate manifest metric etc.

Spline Builder Setup

Design Objective Behind Interpolating Splines

1. <u>Symbols and Definition</u>:
   - Good overview of the desired characteristics is provided in Goodman (2002).
   - Data:

$$\{x_i, y_i\} \in \mathbb{R}^2; i = 0, \cdots, N;\ x_0 < x_1 < \cdots < x_i < \cdots < x_N$$

   - Interpolating function:

$$f(x_i) = y_i; f: [x_0, x_N] \to [\mathbb{R}^2 \to \mathbb{R}]$$

   - Optional Actual Function: $g(x)$
2. <u>Monotonicity</u>: $f(x_i)$ increases with increase in y_i (and vice versa).
   - **Truly monotonic** means that the segment extrema match $g(x)$ extrema.
   - **Co-monotone** => $f(x_i)$ increases with increase in y_i within the segment (and vice versa)
     - Strictly **co-monotone** implies that sub-segment **monotonicity** must also be met, so "**local monotonicity**" where **monotonicity** matches between $f(x_i)$ and y_i at the segment level, is what is accepted – here, there can be an inflection among segments in the immediate proximity of the data extrema.
   - At most, one extrema is allowed in $\{x_i, x_{i+1}\}$.
3. <u>Convexity</u>: $f(x_i)$ should also be convex wherever y_i is convex (and vice versa).
   - At the segment level this becomes **co-convex**. As before strict **co-convexity** is often highly restrictive, so **local convexity** is preferred. The earlier established conditions should also satisfy convexity criteria.
   - Desirable to have at most one inflection in $\{x_i, x_{i+1}\}$.

4. Smoothness: **Smoothness** (related to **shape-preserving**) corresponds to the least curvature. Even C^0 can be "**smooth**", and so can C^k.
5. Locality: **Locality** means that the dependence of $f(x)$ is primarily only on $f(x_{i-1})$ and $f(x_i)$. This is advantageous to schemes that locally modify/insert the points.
6. Approximation Order: **Approximation Order** indicates the smallest polynomial degree by which $f(x)$ departs from $g(x)$ as the density of x increases. More formally, it is the m in

$$\|f - g\| \approx \mathcal{O}(h^m)$$

where

$$\max\{x_{i+1} - x_i : i = 0, \cdots, N - 1\}$$

   - For spline segments where $g(x_0)$ through $g(x_N)$ are specified locally, the first degree of departure should the first degree of non-continuity infinitesimally for both polynomial and non-polynomial splines, i.e., it should be $k + 1$ where the continuity criterion is C^k.
7. Other Desired Criteria:
   - The interpolating proxy $f(x)$ should be able to replicate the target $g(x)$.
   - Fairness – loosely a measure of "pleasing to the eye".
   - Possible $f(x)$ invariance under variate scaling/reflection.
   - Controlled derivative behavior => Small changes in x produce small changes in $f(x)$.
8. Assessment of Monotonicity and Convexity: An individual segment can be assessed to be monotone/convex etc. but from the data PoV, peaks, valleys, and inflection occur only at the knots. These can be assessed only at the span level.

Base Formulation

1. Base Mathematical formulation:

$$y(x) = \sum_{i=0}^{n-1} a_i f_i(x)$$

therefore

$$\frac{d^r y(x)}{dx^r} = \sum_{i=0}^{n-1} a_i \frac{d^r f_i(x)}{dx^r}$$

From known nodes $\{x_0, y_0\}$ and $\{x_1, y_1\}$, we can draw the 2 linear equations for a_i:

$$y_0 = \sum_{i=0}^{n-1} a_i f_i(0)$$

$$y_1 = \sum_{i=0}^{n-1} a_i f_i(1)$$

From known nodal derivatives $\{x_0, y_k(x_0)\}_{k=1}^{r}$, where

$$y_k(x_0) = \left[\frac{d^k y}{dx^k}\right]_{x_0}$$

we can draw the following r linear equations for a_i:

$$y_k(x_0) = \sum_{i=0}^{n-1} a_i \left[\frac{d^r f_i(x)}{dx^r}\right]_{x=x_0}$$

where

$$k \Longrightarrow 1, \cdots, r$$

2. Linearity of Segment Coefficients to the Response Values (y_i): In all the spline formulations, the Jacobian $\frac{\partial c_j}{\partial y_i}$ is constant (i.e., independent of the response values or their nodal derivative inputs).
3. Span Boundary Specification:
   - "Natural" Spline – Energy minimization problem – Second Derivative is Zero at either of the extreme nodes.
   - "Financial" Spline – Second Derivative at the left extreme is zero, but first derivative at the right extreme is zero.
   - Clamped Boundary Conditions:

$$\left[\frac{\partial f}{\partial x}\right]_{x=x_0} = \alpha$$

and

$$\left[\frac{\partial f}{\partial x}\right]_{x=x_N} = \beta$$

   - Not-A-Knot Boundary Conditions:

$$\left[\frac{d^3 f}{dx^3}\right]_{x=x_0} = \left[\frac{d^3 f}{dx^3}\right]_{x=x_1}$$

and

$$\left[\frac{d^3 f}{dx^3}\right]_{x=x_{N-1}} = \left[\frac{d^3 f}{dx^3}\right]_{x=x_N}$$

4. Discrete Segment Mesh vs. Inserted Knots: Inserting knot point is similar to discretizing the segment into multiple grids, with one key difference:
   - Discretization uses the same single spline across all the grid units of the segment.

- Inserted knots introduce additional splines – on between each knot pair.

5. Segment Inelastics: These are effectively the same as shape controller, i.e., the following are the shape controlling inelastic parameter set:
   - Tension σ
   - Number of basis n
   - Continuity C^k
   - Optimizing derivative set order m

References

- Goodman, T. (2002): *Shape Preserving Interpolation by Curves*.

B-Splines

Introduction

1. Motivation: As postulated by De Boor et. al. (see De Boor (2001)), B Splines have a geometric interpolant context – thereby with the correspondingly strong CAD/CAG/curve/surface construction focus. Smoothening occurs as a natural part of this.
    - The B Spline generation scheme has a recurrence-based iterative polynomial generator that admits coinciding control points facilitate and surface construction, with shape-preserving interpolation control thrown in.
    - k^{th} Order B-Spline Interpolant: Higher order B-Splines are defined by the recurrence

$$B_{i,k} = \varepsilon_{i,k} B_{i,k-1} + \left(1 - \varepsilon_{i+1,k}\right) B_{i+1,k-1}$$

where

$$\varepsilon_{i,k} = \frac{t - t_i}{t_{i+k-1} - t_i}$$

and

$$B_{i,1}(t) = X_i(t) = 1$$

if

$$t_i < t < t_{i+1}$$

and

$$B_{i,1}(t) = X_i(t) = 0$$

otherwise. Coinciding knots implies that

$$t_i = t_{i+1}$$

which indicates that

$$B_{i,1}(t) = 0$$

2. Recursive Interpolant Scheme: B Spline formulation is recursively interpolant, i.e., the order k spline is interpolant over the order $k - 1$ splines on nodes i and $i + 1$ - this formulation automatically ensures C^{k-2} nodal continuity.
    - As shown in Figure 4, the left interpolator stretch $[i, i + k - 1]$ contains the interpolator pivot at t_i, and the right interpolator stretch $[i + 1, i + k]$ contains the interpolator pivot at t_{i+1}.
    - $B_{p,q}$ spans all the segments between the nodes $[p, \cdots, q]$.
    - Further, the formulation symmetry between the left pivot at $B_{i,k-1}$ and the right pivot at $B_{i+1,k-1}$ retains the interpolation symmetry – among other things, it is responsible for ensuring the C^{k-2} symmetry.
3. B-Spline Order Relationships: Assuming no coincident knots, the following statements are all EQUIVALENT/TRUE:
    - $n + 1$ knot points.
    - n^{th} order B Spline.
    - Polynomial of degree $n - 1$.
    - Continuity criterion C^{n-2}.
4. Expository Formulation:

$$B_{i,j} = \sum_{i=0}^{j} \alpha_{ij} X_{i+j}$$

$$\alpha_{i0} = \varepsilon_{i,k-1} \cdots \varepsilon_{i,0} = \prod_{j=k-1}^{0} \varepsilon_{i,j}$$

where

$$\varepsilon_{i,j} = \frac{t - t_{i+j-1}}{t_{i+j} - t_{i+j-1}}$$

$$\alpha_{ik} = \prod_{l=1}^{k} [1 - \varepsilon_{i+l-1}]$$

where

$$\varepsilon_{i+l-1} = \frac{t - t_{i+l-1}}{t_{i+l} - t_{i+l-1}}$$

5. <u>Spline Coefficient Partition of Unity</u>: Using

$$B_{i,k} = \sum_{j=0}^{k} \alpha_{ij} X_{i+j}$$

it is easy to show that

$$\sum_{j=0}^{k} \alpha_{ij} = 1$$

This simply follows from the recursive nodal interpolation property.

6. <u>Smoothness Multiplicity Order Linker</u>: # smoothness conditions at knot + the multiplicity at the knot = B-Spline Order.

7. Starting Node de-biasing: Left node is always weighted by $\varepsilon_{i,k}$ in the interpolation scheme, but the left node asymmetry is maintained because the denominator in

$$\varepsilon_{i,j} = \frac{t - t_{i+j-1}}{t_{i+j} - t_{i+j-1}}$$

i.e., $t_{i+j} - t_{i+j-1}$ increases in length.

8. Other Single B-Spline Properties:
    - $B_{i,k}$ is a piece-wise polynomial of degree $< k$ ($k - 1$ if the knots are distinct, lesser if the some of the knots coincide).
    - $B_{i,k}$ is zero outside of $[t_i, t_{i+k})$.
    - $B_{i,k}$ is positive in the open interval $[t_i, \cdots, t_{i+k}]$.

9. Formulation off of Starting Node and Starting Order: Given the starting node i and the starting order k, the contribution to the node $i + m$ (i.e., m nodes after the start) and the order (i.e., n nodes after start) can be "series"ed as

$$B_{i+m,k-n} = \mathcal{N}(i + m, k - n + 1 \to k - n)B_{i+m,k-n+1} + \mathcal{B}(i + m - 1 \to i + m, k - n)B_{i+m-1,k-n}$$

- Nodal B-Spline Recursion Stepper:

$$\mathcal{N}(i + m, k - n + 1 \to k - n) = \mathcal{P}(i + m \to i + m, k - n + 1 \to k - n) = \left[\frac{t - t_{i+m}}{t_{i+m+k-n+1} - t_{i+m}}\right]\left[\frac{t_{i+m+k-n} - t}{t_{i+m+k-n} - t_{i+m+1}}\right]$$

- Spline Order B Spline Recursion Stepper:

$$\mathcal{B}(i + m - 1 \to i + m, k - n) = \mathcal{P}(i + m - 1 \to i + m, k - n \to k - n) = \left[\frac{t - t_{i+m}}{t_{i+m+k-n+1} - t_{i+m}}\right]\left[\frac{t - t_{i+m+1}}{t_{i+m+k-n} - t_{i+m+1}}\right]$$

10. Cardinal B-Spline Knot Sequence: Knot sequence Z => Uniformly spaced knots, simplifying the interpolant/recursive analysis significantly –

$$Z \Longrightarrow \{\cdots, -2, -1, 0, +1, +2, \cdots\}$$

- Also all Cardinal B-Splines of a given order k are translates of each other.
- Cardinal B-Spline Order 2:

Range	$B_{i,2}$	$B_{i+1,2}$
$0 \le t < 1$	t	0
$1 \le t < 2$	$2 - t$	$t - 1$
$2 \le t < 3$	0	$3 - t$

- Cardinal B-Spline Order 3:

$$B_{i,3} = \frac{t}{2} B_{i,2} + \frac{3 - t}{2} B_{i+1,2}$$

Range	$B_{i,3}$	$\frac{\partial B_{i,3}}{\partial t}$
$0 \le t < 1$	$\frac{1}{2} t^2$	t
$1 \le t < 2$	$\frac{1}{2}(-2t^2 + 6t - 3)$	$-2t - 3$
$2 \le t < 3$	$\frac{1}{2}(t - 3)^2$	$t - t$

11. Non-coinciding B Spline Segment Relations:

$$B_{i,1} = X_i = \begin{cases} 1 \; t_i \le t < t_{i+1} \\ 0 \;\; otherwise \end{cases}$$

$$B_{i,2} = \left[\frac{t - t_i}{t_{i+1} - t_i}\right] X_0 + \left[\frac{t_{i+2} - t}{t_{i+2} - t_{i+1}}\right] X_1$$

$$B_{i+1,2} = \left[\frac{t - t_{i+1}}{t_{i+2} - t_{i+1}}\right] X_1 + \left[\frac{t_{i+3} - t}{t_{i+3} - t_{i+2}}\right] X_2$$

$$B_{i,3} = \left[\frac{t - t_i}{t_{i+2} - t_i}\right] B_{i,2} + \left[\frac{t_{i+3} - t}{t_{i+3} - t_{i+1}}\right] B_{i+1,2}$$

Range	$B_{i,2}$	$B_{i+1,2}$	$B_{i,3}$
$t_i \leq t < t_{i+1}$	$\frac{t - t_i}{t_{i+1} - t_i}$	0	$\frac{t - t_i}{t_{i+2} - t_i} \frac{t - t_i}{t_{i+1} - t_i}$
$t_{i+1} \leq t < t_{i+2}$	$\frac{t_{i+2} - t}{t_{i+2} - t_{i+1}}$	$\frac{t - t_{i+1}}{t_{i+2} - t_{i+1}}$	$\frac{t - t_i}{t_{i+2} - t_i} \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} \frac{t - t_{i+1}}{t_{i+2} - t_{i+1}}$
$t_{i+2} \leq t < t_{i+3}$	0	$\frac{t_{i+3} - t}{t_{i+3} - t_{i+1}}$	$\frac{t_{i+3} - t}{t_{i+3} - t_i} \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}}$

12. <u>Bernstein B-Spline Knot Sequence</u>: Considering the knot sequence

$$\mathcal{S} \Rightarrow \{0, \cdots, 0, 1, \cdots, 1\}$$

where $\{0, \cdots, 0\}$ occurs μ times, and $\{1, \cdots, 1\}$ occurs ν times. Let

$$B[\mu, \nu, t] = \frac{(\mu + \nu)!}{\mu!\, \nu!} (1 - t)^{\mu} t^{\nu}$$

for

$$0 \leq t < 1$$

$B[\mu, \nu, t]$ has $\nu - 1$ derivatives at

$$t = 0$$

and $\mu - 1$ derivatives at

$$t = 1$$

This is also referred as ν **smoothness conditions** at $t = 0$ and μ **smoothness conditions** at $t = 1$.

13. B-Spline vs. Spline: B-Spline is just a single basis polynomial that is valid across a set of knots. "**Spline**" is a linear combination of such basis B Splines – i.e., the set of all the $B_{i,k}$'s.
14. Spline Definition:

$$B_{i,3} = \sum_i B_{i,k} a_i$$

where

$$a_i \in \mathbb{R}^1$$

a_i's are the coefficients – or nodal points $\{x_i, a_i\}$ - that can be interpolated.

B Spline Derivatives

1. B-Spline Derivative Formulation:

$$\frac{\partial^r B_{i,k}}{\partial t^r} = \left[\frac{r}{t_{i+k-1} - t_i}\right] \frac{\partial^{r-1} B_{i,k-1}}{\partial t^{r-1}} + \left[\frac{t - t_i}{t_{i+k-1} - t_i}\right] \frac{\partial^r B_{i,k-1}}{\partial t^r} + \left[\frac{r}{t_{i+k} - t_{i+1}}\right] \frac{\partial^{r-1} B_{i+1,k-1}}{\partial t^{r-1}} + \left[\frac{t_{i+k} - t}{t_{i+k} - t_{i+1}}\right] \frac{\partial^r B_{i+1,k-1}}{\partial t^r}$$

2. B Spline Order 3 Nodal Slopes: The slopes match across the left and the right segment, as shown below, thereby making $B_{3,i}$ C^1 continuous.

Range	Left Slope	Right Slope
t_i	-	0
t_{i+1}	$\frac{t_{i+1} - t}{t_{i+2} - t_i}$	$\frac{t_{i+1} - t}{t_{i+2} - t_i}$
t_{i+2}	$\frac{t_{i+3} - t}{t_{i+2} - t_{i+1}}$	$\frac{t_{i+3} - t}{t_{i+2} - t_{i+1}}$
t_{i+3}	0	-

3. <u>B Spline Continuity Condition</u>: From the B Spline derivative formulation it is clear that if both $B_{i,k-1}$ and $B_{i+1,k-1}$ are C^{k-3} continuous, then $B_{i,k}$ will be C^{k-2} continuous. Given that B Spline order 3 is C^1 continuous, by induction, $B_{i,k}$ is C^{k-2} continuous.

References

- De Boor, C. (2001): *Practical Guide to Splines (Revised Edition)* **Springer**.

Polynomial Spline Basis Function

Polynomial Spline Basis Function

1. Most direct polynomial spline fit is the Lagrange polynomial that passes through the sequence of given points.
2. Young (1971) was one of first to apply shape-preserving polynomial using diminished Lagrange Polynomials (Lagrange Polynomials (Wiki)), showing that co-monotone interpolant with an upper bound on the polynomial degree exists (Raymon (1981)).
3. <u>Knot Insertion and Control Techniques</u>: Careful knot insertion can produce:
   - Convexity Preserving Schemes on C^2 cubic (de Boor (2001)).
   - Co-monotone, co-convex schemes on C^1 quadratic (McAllister and Roulier (1981a), McAllister and Roulier (1981b), Schumaker (1983)).
   - Co-monotone on C^1 cubic (Butland (1980), Fritsch and Butland (1984), Fritsch and Carlson (1980), Utreras and Celis (1983)).
   - Co-monotone and co-convex on C^1 cubic (Costantini and Morandi (1984)).
   - C^2 co-monotone and co-convex by using cubic in any interval where there is an inflection, and linear/quadratic rational elsewhere (Schaback (1973), Schaback (1988)).

Bernstein Polynomial Basis Function

1. <u>Bernstein Polynomial of degree n, and term ν</u>:

$$b_{\nu,n}(x) = \frac{(n+\nu)!}{n!\,\nu!}(1-x)^n x^\nu$$

   where

$$\nu = 0, \cdots, n$$

which may be re-written as

$$b_{\nu,n}(x) = n! \frac{x^{\nu}}{\nu!} \frac{(1-x)^{\mu}}{(n-\nu)!}$$

The following re-cast makes it amenable to a recursive formulation:

$$P_{b,c}(x) = (b+c)! \cdot F(x,b) \cdot F(1-x,c)$$

where

$$F(x,b) = \frac{x^b}{b!}$$

2. Derivative of the Bernstein Polynomial:

$$\frac{d^r b_{\nu,n}(x)}{dx^r} = n\left[\frac{d^{r-1} b_{\nu-1,n}(x)}{dx^{r-1}} - \frac{d^r b_{\nu-1,n-1}(x)}{dx^{r-1}}\right]$$

Using the Bernstein Polynomial Re-cast above the derivative becomes

$$\frac{\partial P_{b,c}(x)}{\partial x} = (b+c)!\left[\frac{\partial F(x,b)}{\partial x} \cdot F(1-x,c) + F(x,b) \cdot \frac{\partial F(1-x,c)}{\partial x}\right]$$

3. Bernstein Recurrence:

$$b_{\nu,n}(x) = (1-x) b_{\nu,n-1}(x) + x b_{\nu-1,n-1}(x)$$

4. Reduction of B-Splines to Bernstein's Polynomial: From the recurrence relation, it is clear that this is exactly the same recurrence as that for B-splines, except that it happens over repeating knots at

$$x = 0$$

and

$$x = 1$$

5. Further

$$b_{0,i} = \begin{cases} 1 & 0 \le x < 1 \\ 0 & otherwise \end{cases}$$

References

- Butland, J. (1980): A Method of Interpolating Reasonable-Shaped Curve through any Data *Proc. Computer Graphics* **80** 409-422.
- Costantini, P., and R. Morandi (1984): Monotone and Convex Cubic Spline Interpolation *Calcolo* **21** 281-294.
- De Boor, C. (2001): *Practical Guide to Splines (Revised Edition)* **Springer**.
- Fritsch, F. N., and Carlson, R. E. (1980): Monotone piecewise cubic interpolation *SIAM J. Numerical Anal.* **17** 238-246.
- Fritsch, F. N., and Butland, J. (1984): A method for constructing local monotone piecewise cubic interpolants *SIAM J. Sci. Stat. Comput.* **5** 300-304.
- Goodman, T. (2002): *Shape Preserving Interpolation by Curves*.
- Lagrange Polynomial (Wiki): *Wikipedia Entry for Lagrange Polynomial*.
- McAllister, D., and J Roulier (1981a): An Algorithm for Computing Shape-Preserving Osculating Quadratic Splines *ACM Trans. Math. Software* **7** 331-347.

- McAllister, D., and J Roulier (1981b): Shape-Preserving Osculating Splines *ACM Trans. Math. Software* **7** 384-386.
- Raymon, L. (1981): Piecewise Monotone Interpolation in Polynomial Type *SIAM J. Math. Anal.* **12** 110-114.
- Utreras, F. I., and V. Celis (1983): *Piecewise Cubic Monotone Interpolation: A Variational Approach* **Departamento de Matematicas, Universidad de Chile**, **Tech. Report MA-83-B-281**.
- Schaback, R (1973): Spezielle Rationale Splinefunktionen *J. Approx. Theory* **7** 281-292.
- Schaback, R. (1988): *Adaptive Rational Splines* **NAM-Bericht Nr. 60, Universitat Gottingen**.
- Schumaker, L. L. (1983): On Shape-Preserving Quadratic Spline Interpolation *SIAM J. Numerical Anal* **20** 854-864.
- Young, S. W. (1971): Piecewise Monotone Polynomial Interpolation *Bull. Amer. Math. Soc.* **73** 642-643.

Local Spline Stretches

Local Interpolating/Smoothing Spline Stretches

1. Hermite Cubic Splines: The "local information" here takes the form of user specified left/right slopes + calibration points.
   - 2 User Specified local slopes + 2 points => 4 sets of equations. Solve for the coefficients.
   - C^1 continuity is maintained, and C^2 continuity is not.
   - Segment control is completely local. Both the head and non-head calibration are identical/analogous for this reason.
2. C^1 Hermite Formal Definition: For C^1, the Hermite polynomial of degree $2n+1$ is given as

$$H_{2n+1}(x) = \sum_{j=0}^{n} \{f_j H_{n,j}(x) + f_j{}' \hat{H}_{n,j}(x)\}$$

where $H_{n,j}(x)$ and $\hat{H}_{n,j}(x)$ are expressed in terms of the j[th] Lagrange coefficient of degree n, $L_{n,j}(x)$ as

$$H_{n,j}(x) = \left[1 - 2\left(x - x_j\right)L_{n,j}{}'(x)\right]L_{n,j}{}^2(x)$$

$$\hat{H}_{n,j}(x) = \left(x - x_j\right)L_{n,j}{}^2(x)$$

3. Catmull-Rom Cubic Splines (Catmull and Rom (1974)): Instead of explicitly specifying the left/right segment slopes, they are inferred from the "averages" of the prior and the subsequent points, i.e.

$$\vec{\tau}_i = \frac{1}{2}[\vec{p}_{i+1} - \vec{p}_{i-1}]$$

and

$$\vec{\tau}_{i+1} = \frac{1}{2}[\vec{p}_{i+2} - \vec{p}_i]$$

Here $\vec{\tau}_i$ refers to the slope vector, and $\vec{p}_i$ to the point vector.

- Again, C^1 continuity is maintained, and C^2 continuity is not.
- Segment control is not completely local, but still local enough – it only depends on the neighborhood of 3 points.

4. Cardinal Cubic Splines: This is a generalization of the Catmull-Rom spline with a tightener coefficient σ, i.e.

$$\vec{\tau}_i = \frac{1}{2}(1 - \sigma)[\vec{p}_{i+1} - \vec{p}_{i-1}]$$

and

$$\vec{\tau}_{i+1} = \frac{1}{2}(1 - \sigma)[\vec{p}_{i+2} - \vec{p}_i]$$

$\sigma > 0$ corresponds to tightening, and $\sigma < 0$ corresponds to loosening.

- Again, C^1 continuity is maintained, and C^2 continuity is not.
- Segment control is "local" in the Catmull-Rom sense - it only depends on the neighborhood of 3 points.

5. Catmull-Rom/Cardinal Splines as Interpolation Splines: As interpolating splines, both Catmull-Rom and Cardinal are primarily useful in heuristic knot-insertions – Catmull-Rom as linear in the gaps, and Cardinal as tense linear gap knots.

- The local knot point insertion may be generalized as follows: The targeted knot insertions follow the formulation paradigm

$$f(x_{KNOT}) = f(x_i : i \,\forall\, \aleph)$$

where $\aleph$ is the set of the neighborhood points. Similar formulation (with potentially different function forms, of course) may be used for each of the C^k derivatives. Catmull-Rom and cardinal use 1D, strictly neighboring adjacencies, as well as tense linear averaging.

6. C^1 Hermite-Bessel Splines: These splines use 4 basis functions per segment, therefore they are cubic polynomial, but C^1. The first are set at each node x_i as the first derivative of the quadratic that passes through x_{i-1}, x_i, and x_{i+1} (the edges are handled slightly differently, as shown below). Specifically

$$b_0 = \frac{1}{x_2 - x_0}\left[\frac{(x_2 + x_1 - 2x_0)(y_1 - y_0)}{x_1 - x_0} - \frac{(x_1 - x_0)(y_2 - y_1)}{x_2 - x_1}\right]$$

$$b_n = \frac{1}{x_n - x_{n-2}}\left[\frac{(x_n - x_{n-1})(y_{n-1} - y_{n-2})}{x_{n-1} - x_{n-2}} - \frac{(2x_n - x_{n-1} - x_{n-2})(y_n - y_{n-1})}{x_n - x_{n-1}}\right]$$

$$b_i = \frac{1}{x_{i+1} - x_{i-1}}\left[\frac{(x_{i+1} - x_i)(y_i - y_{i-1})}{x_i - x_{i-1}} + \frac{(x_i - x_{i-1})(y_{i+1} - y_i)}{x_{i+1} - x_i}\right] \forall\, 1 \le i < n-1$$

7. Hyman's Monotone Preserving Cubic Spline:
   - Hyman (1983) applies the stringent conditions to preserve monotonicity by applying the de Boor-Schwarz criterion.
   - Define

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

If locally monotone, i.e.

$$m_{i-1} m_i \ge 0$$

then set

$$b_i = \frac{3m_{i-1}m_i}{\max(m_{i-1}, m_i) + 2 \cdot \min(m_{i-1}, m_i)}$$

If non-monotone, i.e.

$$m_{i-1}m_i < 0$$

then set

$$b_i \geq 0$$

- Put another way (Iwashita (2013)): For cubic polynomial splines, the first derivative should be in the range

$$-\frac{3\tau_i f_i}{h_i} \leq \tau_i {f_i}' \leq \frac{3\tau_i f_i}{h_{i-1}}$$

where

$$\tau_i \Longrightarrow sign(f_i)$$

- Adjustment for Spurious Extrema => To ensure that no spurious extrema is introduced in the interpolant

$$b_{i,Adj} = \begin{cases} \min(\max(0, b_i), 3 \cdot \min(m_{i-1}, m_i)) \ y_{i+1} > y_i \\ \max(\min(0, b_i), 3 \cdot \max(m_{i-1}, m_i)) \ y_{i+1} < y_i \end{cases}$$

- <u>Hyman89 Extension to Hyman83</u>: Doherty, Edelman, and Hyman (1989) relax the Hyman83 stringency posited for monotonicity preservation. Define the following:

$$\mu_{i,-1} = \frac{m_i[2(t_i - t_{i-1}) + t_{i-1} - t_{i-2}] + m_{i-1}(t_i - t_{i-1})}{t_i - t_{i-2}}$$

$$\mu_{i,0} = \frac{m_{i-1}(t_{i+1} - t_i) + m_i(t_i - t_{i-1})}{t_{i+1} - t_{i-1}}$$

$$\mu_{i,1} = \frac{m_i[2(t_{i+1} - t_i) + t_{i+2} - t_{i+1}] + m_{i+1}(t_{i+1} - t_i)}{t_{i+2} - t_i}$$

$$M_i = 3 \cdot \min\left(|m_{i-1}|, |m_i|, |\mu_{i,0}|, |\mu_{i,1}|\right)$$

- If $i > 2$, $\mu_{i,-1}$, $\mu_{i,0}$, $m_{i-1} - m_{i-2}$, and $m_i - m_{i-1}$ all have the same sign, then

$$M_i = \max\left[M_i, 1.5 \cdot \min\left(|\mu_{i,0}|, |\mu_{i,-1}|\right)\right]$$

- If $i > n - 1$, $-\mu_{i,0}$, $-\mu_{i,1}$, $m_i - m_{i-1}$, and $m_{i+1} - m_i$ all have the same sign, then

$$M_i = \max\left[M_i, 1.5 \cdot \min\left(|\mu_{i,0}|, |\mu_{i,-1}|\right)\right]$$

- Finally, set

$$s_i = \begin{cases} sign(s_i) \cdot \min(M_i, |s_i|) & sign(s_i) = sign\left(\mu_{i,0}\right) \\ s_i = 0 & otherwise \end{cases}$$

8. <u>Hyman's Monotone Preserving Quintic Spline</u>:
    - For quintic polynomial splines, the first derivative should be in the range

$$-\frac{5\tau_i f_i}{h_i} \le \tau_i f_i' \le \frac{5\tau_i f_i}{h_{i-1}}$$

where

$$\tau_i \Longrightarrow sign(f_i)$$

- The constraint on the second derivative is

$$\tau_i f_i'' \leq \tau_i \cdot \max\left(8\frac{f_i'}{h_{i-1}} - 20\frac{f_i}{{h_{i-1}}^2}, -8\frac{f_i'}{h_{i-1}} - 20\frac{f_i}{{h_{i-1}}^2}\right)$$

- Monotonicity Preserving Quintic Spline => Enhancement of the criterion established by de Boor and Schwartz (1977) (Hyman (1983), Doherty, Edelman, and Hyman (1989)). Set

$$\sigma_i = sign(f_i)$$

if

$$s_{i-1}s_i < 0$$

and

$$\sigma_i = 0$$

otherwise. If

$$\sigma_i \geq 0$$

then

$$f_i' = \min(\max(0, f_i'), 5 \cdot \min(|s_{i-1}|, |s_i|))$$

If

$$\sigma_i \leq 0$$

then

$$f_i' = \max(\min(0, f_i'), -5 \cdot \min(|s_{i-1}|, |s_i|))$$

- Second Derivative Tests for Monotonicity Preserving Quintic Spline => Define the following constants

$$a = \max\left(0, \frac{f_i'}{s_{i-1}}\right)$$

$$b = \max\left(0, \frac{f_{i+1}'}{s_i}\right)$$

$$d_+ = \begin{cases} f_i' & f_i' s_i > 0 \\ 0 & otherwise \end{cases}$$

$$d_- = \begin{cases} f_i' & f_i' s_{i-1} > 0 \\ 0 & otherwise \end{cases}$$

Define Ranges A and B as

$$A = \left[\frac{-7.9d_+ - 0.26bd_+}{h_i}, \frac{(20 - 2b)s_i - 8d_+ - 0.48ad_+}{h_i}\right]$$

$$B = \left[\frac{(20 - 2a)s_{i-1} + 8d_- + 0.48ad_-}{h_{i-1}}, \frac{7.9d_- + 0.26bd_-}{h_{i-1}}\right]$$

If A and B overlap, then f_i'' should lie in their common range. If they do not overlap, i.e., if

$$A \cap B = \emptyset$$

reset f_i' as

$$f_i{}' = \frac{\frac{(20-2b)s_i}{h_i} + \frac{(20-2a)s_{i-1}}{h_{i-1}}}{\frac{8+0.48b}{h_i} + \frac{8+0.48a}{h_{i-1}}}$$

Setting this $f_i{}'$ ensures that A and B overlap, so the other tests aspects may now continue.

9. Harmonic Splines: These were introduced by Fritsch and Butland (1984) as

$$\frac{1}{s_i} = \frac{t_i - t_{i-1} + 2(t_{i+1} - t_i)}{3(t_{i+1} - t_{i-1})}\frac{1}{m_{i-1}} + \frac{t_{i+1} - t_i + 2(t_i - t_{i-1})}{3(t_{i+1} - t_{i-1})}\frac{1}{m_i}$$

if

$$m_{i-1}m_i > 0$$

and

$$s_i = 0$$

if

$$m_{i-1}m_i \le 0$$

The boundary conditions are

$$s_0 = \frac{t_2 - t_1 + 2(t_1 - t_0)}{t_2 - t_0}m_0 - \frac{t_1 - t_0}{t_2 - t_0}m_1$$

$$s_n = \frac{t_{n-1} - t_{n-2} + 2(t_n - t_{n-1})}{t_n - t_{n-2}}m_{n-1} - \frac{t_n - t_{n-1}}{t_n - t_{n-2}}m_{n-2}$$

- Harmonic Spline Monotonicity Filter =>

$$s_0 = \begin{cases} 0 & s_0 m_0 \le 0 \\ 3m_0 & s_0 m_0 > 0;\ m_0 m_1 \le 0;\ |s_0| < |3m_0| \\ s_0 & s_0 m_0 > 0 \end{cases}$$

$$s_n = \begin{cases} 0 & s_n m_{n-1} \le 0 \\ 3m_{n-1} & s_n m_{n-1} > 0;\ m_{n-1} m_{n-2} \le 0;\ |s_n| < |3m_{n-1}| \\ s_n & s_n m_{n-1} > 0 \end{cases}$$

- Continuous Limiters => For harmonic splines, as the predictor ordinate widths become identical, i.e.,

$$t_i - t_{i-1} \to t_{i+1} - t_i$$

setting

$$r = \frac{m_i}{m_{i-1}}$$

we get

$$s_i = m_i \frac{r + |r|}{1 + |r|}$$

This is the Van Leer limit (Van Leer (1974)). Huynh (1993) reviews several such limiters.

- Shortcoming of these Limiters => Since they rely on min/max/modulus functions, by definition they are not smooth close to transition edge. This is rectified by Le Floc'h (2013), who defines a new limiter based on rational functions:

$$s_i = \frac{3m_i m_{i-1}(m_i + m_{i-1})}{{m_i}^2 + 4m_i m_{i-1} + {m_{i-1}}^2}$$

for

$$m_i m_{i-1} > 0$$

and

$$s_i = 0$$

otherwise. This produces a stable C^1 interpolator.

10. Akima Cubic Interpolator (Akima (1970)):
    - Expand the Predictor Ordinates => Add 2 predictor ordinates each at the left and the right boundaries using $x_{-2}, x_{-1}, x_0, \cdots, x_N, x_{N+1}, x_{N+2}$:

$$x_{N+2} - x_N = x_{N+1} - x_{N-1} = x_N - x_{N-2}$$

$$x_{-2} - x_0 = x_{-1} - x_1 = x_0 - x_2$$

    - Expand the Response Values => Add 2 response values each at the left and the right boundaries using $y_{-2}, y_{-1}, y_0, \cdots, y_N, y_{N+1}, y_{N+2}$ using

$$\frac{y_{N+2} - y_{N+1}}{x_{N+2} - x_{N+1}} - \frac{y_{N+1} - y_N}{x_{N+1} - x_N} = \frac{y_{N+1} - y_N}{x_{N+1} - x_N} - \frac{y_N - y_{N-1}}{x_N - x_{N-1}} = \frac{y_N - y_{N-1}}{x_N - x_{N-1}} - \frac{y_{N-1} - y_{N-2}}{x_{N-1} - x_{N-2}}$$

$$\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0} - \frac{y_0 - y_{-1}}{x_0 - x_{-1}} = \frac{y_0 - y_{-1}}{x_0 - x_{-1}} - \frac{y_{-1} - y_{-2}}{x_{-1} - x_{-2}}$$

    - Final C^1 Slope => Compute the final Akima C^1 slopes using

$$f_i{}' = \frac{|s_{i+1} - s_i| s_{i-1} + |s_{i-1} - s_{i-2}| s_i}{|s_{i+1} - s_i| + |s_{i-1} - s_{i-2}|}$$

    if

$$s_{i+1} \neq s_i$$

or

$$s_{i-1} \neq s_{i-2}$$

Otherwise

$$f_i{}' = \frac{s_{i+1} + s_i}{2}$$

11. <u>Kruger's Constrained Cubic Interpolant (Kruger (2002))</u>: If

$$s_i s_{i-1} > 0$$

then

$$f_i{}' = 0$$

Otherwise

$$\frac{2}{f_i{}'} = \frac{1}{s_{i-1}} + \frac{1}{s_i}$$

At the end points

$$f_0{}' = \frac{3}{2} s_0 - \frac{1}{2} f_1{}'$$

$$f_N{}' = \frac{3}{2} s_{N-1} - \frac{1}{2} f_{N-1}{}'$$

12. <u>Shape-Preserving Knot-based C^2 Cubic</u>: Ideas are taken from the awesome paper by Pruess (1993). The basic idea is to take the interval $[x_i, x_{i+1}]$, and partition it into 3 parts by locating the two knots at

$$\xi_i = x_i + \sigma_i h_i$$

and

$$\eta_i = x_{i+1} - \tau_i h_i$$

Evolve the criteria for the selection of ξ_i and η_i (and, of course, their corresponding responses) such that the local spline has shape-preserving feature, and avoids being global (i.e., preserves locality). Using these notations, the basic equations are

$$f(x) = f(x_i) + [x - x_i]f'(x_i) + \frac{1}{2}[x - x_i]^2 f''(x_i) + \frac{1}{6}[x - x_i]^3 f'''(x_i)\ \forall\ x \in\ [x_i, \xi_i]$$

$$f(x) = f(\xi_i) + [x - \xi_i]f'(\xi_i) + \frac{1}{2}[x - \xi_i]^2 f''(\xi_i) + \frac{1}{6}[x - \xi_i]^3 f'''(\xi_i)\ \forall\ x \in\ [\xi_i, \eta_i]$$

$$f(x) = f(\eta_i) + [x - \eta_i]f'(\eta_i) + \frac{1}{2}[x - \eta_i]^2 f''(\eta_i) + \frac{1}{6}[x - \eta_i]^3 f'''(\eta_i)\ \forall\ x \in\ [\eta_i, x_{i+1}]$$

The corresponding C^2 maintenance solution then becomes (in terms of $f'(x_i)$, $f''(x_i)$, σ_i, and τ_i, whose specification will then complete the inference):

$$f(\xi_i) = f(x_i) + \sigma_i h_i f'(x_i) + \frac{{\sigma_i}^2 {h_i}^2}{6}\{2f''(x_i) + f''(\xi_i)\}$$

$$f(\eta_i) = f(x_{i+1}) + \tau_i h_i f'(x_{i+1}) + \frac{{\tau_i}^2 {h_i}^2}{6}\{2f''(x_{i+1}) + f''(\eta_i)\}$$

$$f'(\xi_i) = f'(x_i) + \frac{\sigma_i h_i}{2}\{f''(x_i) + f''(\xi_i)\}$$

$$f'(\eta_i) = f'(x_{i+1}) - \frac{\tau_i h_i}{2}\{f''(x_{i+1}) + f''(\eta_i)\}$$

$$f''(\xi_i) = \frac{6s_i - 2(2 + \sigma_i - \tau_i)f'(x_i) - 2(1 - \sigma_i + \tau_i)f'(x_{i+1}) - \sigma_i h_i(2 - \tau_i)f''(x_i) + \tau_i h_i(1 - \sigma_i)f''(x_{i+1})}{h_i(1 - \tau_i)}$$

$$f''(\eta_i) = \frac{-6s_i + 2(1 + \sigma_i - \tau_i)f'(x_i) + 2(2 - \sigma_i + \tau_i)f'(x_{i+1}) + \sigma_i h_i(1 - \tau_i)f''(x_i) - \tau_i h_i(2 - \sigma_i)f''(x_{i+1})}{h_i(1 - \sigma_i)}$$

$$f'''(x_i) = \frac{f''(\xi_i) - f''(x_i)}{\sigma_i h_i}$$

$$f'''(\xi_i) = \frac{f''(\eta_i) - f''(\xi_i)}{(1 - \sigma_i - \tau_i)h_i}$$

$$f'''(\eta_i) = \frac{f''(x_{i+1}) - f''(\eta_i)}{\tau_i h_i}$$

- Choice of $f'(x_i)$ and $f''(x_i)$: $f'(x_i)$ and $f''(x_i)$ may be generated using typical generation schemes (e.g., using the Fritsch and Butland (1984) algorithm).
- The Preuss Inequalities: It is specified as follows. Set

$$\beta_i = sign(s_i - s_{i-1})$$

$$R_{2i-1} = 6s_i - 4{f_i}' - 2{f_{i+1}}'$$

$$R_{2i} = 2{f_i}' + 4{f_{i+1}}' - 6s_i$$

If

$$\beta_i R_{2i} \geq 0$$

and

$$\beta_i R_{2i-1} \geq 0$$

you are done, since the chosen ${f_i}'$ and ${f_i}''$ also preserve convexity – you can go and set the second derivatives, and set τ_i and σ_i.

- Mismatch in the Preuss Inequalities: If the Preuss inequalities are not met, ${f_i}'$ and ${f_i}''$ need to be modified such that

$$ {f_i}' \in [c_i, d_i] $$

where c_i and d_i are obtained using the double sweep algorithm below.

- Preuss (1993) Double Sweep: First find a_i and b_i from the following regimes:

$$a_i = \begin{cases} \max\left(s_i, \dfrac{3s_i - b_i}{2}\right) & \beta_i > 0, \beta_{i+1} > 0 \\ \max\left(3s_i - b_i, \dfrac{3s_i - b_i}{2}\right) & \beta_i < 0, \beta_{i+1} > 0 \\ \max(s_{i+1}, 3s_i - b_i) & \beta_i > 0, \beta_{i+1} < 0 \end{cases}$$

$$b_i = \begin{cases} 3s_i - a_i & \beta_i > 0, \beta_{i+1} > 0 \\ \max\left(3s_i - 2a_i, \dfrac{3s_i - a_i}{2}\right) & \beta_i < 0, \beta_{i+1} > 0 \\ \dfrac{3s_i - a_i}{2} & \beta_i > 0, \beta_{i+1} < 0 \end{cases}$$

- Finally the a_0 and b_0 initializations are set as follows:

$$a_0 = \begin{cases} 3s_0 - 2b_0 & \beta_0 > 0, \beta_1 > 0 \\ s_0 & \beta_0 < 0, \beta_1 > 0 \end{cases}$$

$$b_0 = \begin{cases} s_0 & \beta_0 > 0, \beta_1 > 0 \\ 3s_0 - 2s_1 & \beta_0 < 0, \beta_1 > 0 \end{cases}$$

- Preuss (1993) Backward Sweep for c_i, d_i: First set

$$s_N{}' = f_N{}' = \frac{a_N + b_N}{2}$$

then set c_i, d_i using the following:

$$c_i = \begin{cases} 3s_i - f_{i+1}{}' & \beta_i > 0, \beta_{i+1} > 0 \\ \max\left(3s_i - 2f_{i+1}{}', \dfrac{3s_i - f_{i+1}{}'}{2}\right) & \beta_i < 0, \beta_{i+1} > 0 \\ \dfrac{3s_i - f_{i+1}{}'}{2} & \beta_i < 0, \beta_{i+1} < 0 \end{cases}$$

$$d_i = \begin{cases} \dfrac{3s_i - f_{i+1}{}'}{2} & \beta_i > 0, \beta_{i+1} > 0 \\ \max\left(3s_i - 2f_{i+1}{}', \dfrac{3s_i - f_{i+1}{}'}{2}\right) & \beta_i < 0, \beta_{i+1} > 0 \\ 3s_i - f_{i+1}{}' & \beta_i < 0, \beta_{i+1} < 0 \end{cases}$$

- Preuss (1993) Setting 2nd Derivatives:

$$f_i{}'' = \beta_i \left[\frac{\beta_i R_{2i-1}}{h_i}, \frac{\beta_i R_{2i-2}}{h_{i-1}}\right]$$

- Preuss (1993) Final Step – Setting τ_i and σ_i: Setting

$$\tau_i = \sigma_i = \frac{1}{3}$$

verify the following inequalities:

$$\beta_i[4f_i' + 2f_i'' + h_i f_i'' \tau_i(2-\tau_i) - h_i f_{i+1}'' \tau_i(1-\tau_i)] \le 6\beta_i s_i$$

$$\beta_{i+1}[2f_i' + 4f_i'' + h_i f_i'' \tau_i(1-\tau_i) - h_i f_{i+1}'' \tau_i(2-\tau_i)] \le 6\beta_{i+1} s_i$$

If these inequalities are satisfied, then you have your f_i', f_i'', τ_i, and σ_i. Otherwise, reduce τ_i and σ_i till they are satisfied.

Space Curves and Loops

1. Space Curve Reproduction: Here is one way to construct loops that are not possible using the ordered variates, i.e., $x_1 < x_2 < \cdots < x_n$.
    - If the ordering $x_1 < x_2 < \cdots < x_n$ is switched out in favor of the DAG $\{x_j, y_j\}$, where the DAG vertices correspond to the loop trace, normal splines may be used to represent space curve loops.
2. Second Degree Parameterization: However, on using a second-degree parameterization of x and y such as

$$x = f_X(u)$$

and

$$y = f_y(u)$$

it may be possible to enforce the order $u_1 < u_2 < \cdots < u_n$. The corresponding control points are $[u_1, x_1], \cdots, [u_n, x_n]$ and $[u_1, y_1], \cdots, [u_n, y_n]$.

   a. Side effect of this – is that you need to work on a pair of splines – one each for

$$x = f_X(u)$$

and

$$y = f_y(u)$$

b. This can offer additional customization and freedom in the design of the surface, at the expense of computing additional splines.

3. Closed Loops: Further, if the start/end points coincide, this corresponds to a closed loop that satisfies the C^2 continuity criterion.
    a. This also implies that no extra head/tail C^1 slope specifications are required.

References

- Akima, H. (1970): A New Method of Interpolation and Smooth Curve Fitting based on Local Procedures *J. Association for Computing Machinery* **17 (4)** 589-602.
- Catmull, E., and R. Rom (1974): A Class of local Interpolating Spline, in: *Computer Aided Geometric Design* R. E. Barnhill and Reisenfled (Eds.) **Academic Press** New York.
- De Boor, C. and B. Schwartz (1977): Piecewise Monotone Interpolation *Journal of Approximation Theory* **21** 411-416.
- Dougherty, R., A. Edelman, and J. Hyman (1989): Non-negativity, Monotonicity, and Convexity Preserving Cubic and Quintic Hermite Interpolation *Mathematics of Computation* **52 (186)** 471-494.
- Fritsch, F. N., and Butland, J. (1984): A Method for Constructing Local Monotone Piecewise Cubic Interpolants *SIAM J. Sci. Stat. Comput.* **5** 300-304.
- Huynh, H. (1993): Accurate Monotone Cubic Interpolation *SIAM Journal on Numerical Analysis* **30 (1)** 57-100.
- Hyman, J. M. (1983): Accurate Monotonicity Preserving Cubic Interpolation *SIAM Journal of Scientific and Statistical Computing* **4 (4)** 645-654.
- Iwashita, Y. (2013): *Piecewise Polynomial Interpolations* Open Gamma Technical Report.

- Kruger, C. J. C. (2002): *Constrained Cubic Spline Interpolation for Chemical Engineering Applications*.
- Le Floc'h, F. (2013): Stable Interpolation for the Yield Curve *Calypso Technology Working Paper Series*.
- Pruess, S. (1993): Shape Preserving C^2 Cubic Spline Interpolation *IMA Journal of Numerical Analysis* **13 (4)** 493-507.
- Van Leer, B. (1974): Towards the Ultimate Conservative Difference Scheme – II: Monotonicity and Conservation combined in a Second Order Scheme *Journal of Computational Physics* **14 (4)** 361-370.

Spline Segment Calibration

Introduction

1. Spline Segment Calibrator: For a given segment, its calibration depends only on the segment local value set – the other inputs come from the prior segments (except in the case of "left-most" segment, whose full set of inputs will have to be extraneously specified).
2. Bayesian Techniques in Spline Calibration: Frequentist and Bayesian techniques such as MLE and MAP regression ought to be possible in the calibration of the spline segment/span coefficients.
3. Main Calibration Inputs Modes: Here we consider the following segment calibration input modes: a) Smoothing Best fit Splines, and b) Segment Best Fit Response Inputs with Constraints.

Smoothing Best Fit Splines

1. Definition: Here the treatment is limited to within a segment. In this, the segment coefficients are calibrated to the following inputs:
   - Truthness Constraints.
   - Smoothness C^k.
   - Penalizing Segment Smoother.
   - Penalizing Segment Weighted Fitness Match (in the least-squared sense).
2. Nomenclature:
   - $p: \ 0, \cdots, q-1$=> Weighted fitness penalizer match points
   - $\begin{matrix} i: \ 0, \cdots, n-1 \\ j: \ 0, \cdots, n-1 \end{matrix}$ => Segment Basis Functions
   - m => Roughness Penalty Match Derivative Order
   - r => r Separation
   - k => C^k Continuity

3. Spline Set Setup:

$$Gross\ Penalizer = \ Best\ Fit\ Penalizer + Curvature\ Penalizer$$

$$\mathcal{R} = \frac{1}{q}\mathcal{R}_F + \lambda\mathcal{R}_c$$

$$\mathcal{R}_F = \sum_{p=0}^{q-1} W_P\left(y_p - Y_p\right)^2$$

$$\mathcal{R}_c = \int_{x_l}^{x_{l+1}} \left(\frac{\partial^m y}{\partial x^m}\right)^2 dx$$

The spline setup maybe represented as

$$y = \sum_{i=0}^{n-1} \beta_i f_i(x)$$

$$y_p = \sum_{i=0}^{n-1} \beta_i f_i\left(x_p\right)$$

$$\frac{\partial^m y}{\partial x^m} = \sum_{i=0}^{n-1} \beta_i \frac{\partial^m f_i(x)}{\partial x^m}$$

4. Best Fit Penalizer Setup: The first derivative is

$$\frac{\partial\mathcal{R}_F}{\partial\beta_r} = 2\sum_{p=0}^{q-1} W_p f_r\left(x_p\right)\left\{\sum_{i=0}^{n-1} \beta_i f_i\left(x_p\right) - Y_p\right\} = 0$$

The second derivative is

$$\frac{\partial^2 \mathcal{R}_F}{\partial {\beta_r}^2} = 2\sum_{p=0}^{q-1} W_p {f_r}^2(x_p) > 0$$

so $\mathcal{R}_F$ has a minimum.

5. Curvature Penalizer Setup: The first derivative is

$$\frac{\partial \mathcal{R}_c}{\partial \beta_r} = \sum_{i=0}^{n-1} \beta_i \int_{x_l}^{x_{l+1}} \left(\frac{\partial^m f_i}{\partial x^m}\right)\left(\frac{\partial^m f_r}{\partial x^m}\right) dx = 0$$

and the second derivative is

$$\frac{\partial^2 \mathcal{R}_c}{\partial {\beta_r}^2} = 2\int_{x_l}^{x_{l+1}} \left(\frac{\partial^m f_r}{\partial x^m}\right)^2 dx = 0$$

so $\mathcal{R}_c$ has a minimum.

6. Second Derivative:

$$\frac{\partial^2 \mathcal{R}}{\partial {\beta_r}^2} = \frac{1}{q}\frac{\partial^2 \mathcal{R}_F}{\partial {\beta_r}^2} + \lambda \frac{\partial^2 \mathcal{R}_c}{\partial {\beta_r}^2} = 2\left\{\frac{1}{q}\sum_{p=0}^{q-1} W_p {f_r}^2(x_p) + \lambda \int_{x_l}^{x_{l+1}} \left(\frac{\partial^m f_r}{\partial x^m}\right)^2 dx\right\} > 0$$

7. Joint Linearized Minimizer Setup:

$$\frac{1}{2}\frac{\partial \mathcal{R}}{\partial \beta_r} = \left\{\frac{1}{q}\sum_{p=0}^{q-1} W_p f_i(x_p) f_r(x_p) + \lambda \int_{x_l}^{x_{l+1}} \left(\frac{\partial^m f_i}{\partial x^m}\right)\left(\frac{\partial^m f_r}{\partial x^m}\right) dx\right\} - \frac{1}{q}\sum_{p=0}^{q-1} W_p f_r(x_p) Y_p = 0$$

8. Number of Equations/Unknowns Review:

a. For intermediate segments, the following equations determine the unknowns:
    i. Number of Continuity Constraints => k
    ii. Number of Left/Right Node Values => 2
    iii. Number of Roughness Penalizer Constraint => at least 1
    iv. Thus, minimal number of degrees of freedom on a per-segment basis: $k + 3$. This will be the number of "free" parameters we will use for to extract for each segment.
b. For left most segments, the following equations determine the unknowns:
    i. Number of Left/Right Node Values => 2
    ii. Number of Roughness Penalizer Constraint => at least 1
    iii. Thus, for the set of $k + 3$ parameters, the number of undetermined parameters: k
c. For the span as a whole, the number of degrees of freedom/undetermined parameters is k. You may determine:
    i. The right-most second derivative, AND
    ii. Possibly, the left-most second derivative
    iii. For k, this will complete the set of undetermined coefficients.

Segment Best Fit Response with Constraint Matching

1. Purpose: Here we assume that a linear transform exists the hidden state quantification metric and the measurement manifest metric.
2. Caveat with the Segment-wise Representation: Optimizing on certain constraints (such as multi-segment constraints) now ends up producing a highly non-sparse, dense matrix. This is simply a reflection on the multi-segment spanning nature of the constraint and the eventual optimization.
3. Constraint and Least-Square Spec:

$$S_p = \hat{C}_p - C_p$$

where

$$\hat{C}_p = \sum_{s=0}^{t_p - 1} \gamma_s \hat{y}_s$$

and

$$\hat{y}_s = \sum_{i=0}^{n-1} \beta_i f_i(x_s)$$

Note that when the hidden state quantification metric is identically the same as the measurement manifest metric

$$\gamma_s = 1$$

and

$$t_p = 1$$

This corresponds to computing the least-square minimization over the observations.

4. <u>Constraint Formulation Development</u>:

$$\hat{C}_p = \sum_{s=0}^{n-1} \beta_i \sum_{s=0}^{t_p - 1} \gamma_s f_i(x_s)$$

or

$$\hat{C}_p = \sum_{s=0}^{n-1} \beta_i G_{ip}$$

where

$$G_{ip} = \sum_{s=0}^{t_p - 1} \gamma_s f_i(x_s)$$

The parallel between this and the original least-squares formulation can now be extended in a straightforward manner.

5. Weighted Constrained LSM:

$$S_p = \hat{C}_p - C_p = \sum_{i=0}^{n-1} \beta_i G_{ip} - C_p$$

$${S_p}^2 = {\beta_k}^2 {G_{kp}}^2 + 2\beta_k G_{kp} \left[\sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i G_{ip} - C_p \right] + \left[\sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i G_{ip} - C_p \right]^2$$

$$S^2 = \frac{1}{N} \sum_{p=0}^{q-1} W_p {S_p}^2$$

6. Optimization of S^2:

$$S^2 = \frac{1}{N} \left\{ {\beta_k}^2 \sum_{p=0}^{q-1} W_p {G_{kp}}^2 + 2\beta_k \sum_{p=0}^{q-1} W_p G_{kp} \left[\sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i G_{ip} - C_p \right] + \sum_{p=0}^{q-1} W_p \left[\sum_{\substack{i=0 \\ i \neq k}}^{n-1} \beta_i G_{ip} - C_p \right]^2 \right\}$$

$$\frac{\partial (S^2)}{\partial \beta_k} = 0$$

implies that

$$\sum_{i=0}^{n-1} \beta_i \sum_{p=0}^{q-1} W_p G_{ip} G_{kp} = \sum_{p=0}^{q-1} W_p G_{kp} C_p$$

$$\frac{\partial^2 (S^2)}{\partial {\beta_k}^2} = \sum_{p=0}^{q-1} W_p {G_{kp}}^2 \geq 0$$

for

$$W_p \geq 0$$

thus the extremum is a minimum.

Spline Jacobian

Introduction

1. Chain Rule vs. Matrix Operations of Linear Basis Function Combination: When it comes to extracting variate Jacobian of coefficients from boundary inputs, these are absolutely equivalent – in fact, the coefficient Matrix is in reality a Jacobian itself.
   - Matrix entry as a Jacobian => Every entry of the Matrix A where

$$AX = Y$$

   is actually a Jacobian entry, i.e.,

$$A_{ij} = \frac{\partial Y_i}{\partial X_j}$$

2. Self-Jacobian: Given an ordered pair $\{x_i, y_i\}$ that needs to be interpolated/splined across, the self-Jacobian is defined as the vector $\frac{\partial y(x)}{\partial y(x_j)}$. More generally, the self-Jacobian may be defined as $\frac{\partial y(x)}{\partial I_j}$ where I_j is an input.
   - Self-Jacobian tells you the story of sensitivity/perturbability of the interpolant y on non-local points through I_j, since the C^k transmission occurs through v. Within a single segment, higher order splines cause fairly non-banded, dispersed Jacobians, indicating that the impact is non-local; local splines produce simple banded/tri-diagonal Jacobians; and tension splines produce a combination of the two depending on the tension parameter (and therefore dense within the segment).
   - Obviously Jacobian of any function $F\left(y(x)\right)$ is going to be dependent on the self-Jacobian $\frac{\partial y(x)}{\partial I_j}$ because of the chain rule.

Optimizing Spline Basis Function Jacobian

1. Coefficient- Value Micro-Jacobian:

$$FA = Y$$

where A is the matrix of the basis coefficients

$$\{a_0, a_1, a_2, \cdots, a_r, \cdots, a_{k+1}, \cdots, a_{n-1}\}$$

Y is the matrix (column valued) of the values (RHS). In particular, it is the boundary segment calibration nodal values in the following order:

$$\left\{y_0, y_1, \left\|\frac{\partial y}{\partial x}\right\|_{x=0}, \cdots, \left\|\frac{\partial^r y}{\partial x^r}\right\|_{x=0}, \cdots, \left\|\frac{\partial^k y}{\partial x^k}\right\|_{x=0}, 0, \cdots, 0\right\}$$

F is the matrix of the coefficients of the basis function values and their derivatives. Its contents are as follows. For

$$l = 0; j = 0, \cdots, n-1$$

we get

$$F_{0j} = f_j(x = 0)$$

For

$$l = 1; j = 0, \cdots, n-1$$

we get

$$F_{1j} = f_j(x = 1)$$

For

$$2 \leq l \leq k + 1; j = 0, \cdots, n - 1$$

we get

$$F_{lj} = \left[\frac{\partial^{l-1} f_j}{\partial x^{l-1}}\right]_{x=0}$$

and finally for

$$l = k + 2, \cdots, n - 1; j = 0, \cdots, n - 1$$

we get

$$F_{lj} = Q_{lj}$$

where

$$Q_{lj} = \int_{x_l}^{x_{l+1}} \left(\frac{\partial^m f_i}{\partial x^m}\right)\left(\frac{\partial^m f_j}{\partial x^m}\right) dx$$

2. <u>Coefficient-Value Micro-Jacobian</u>: Given

$$FA = Y$$

the coefficient-value micro-Jack is

$$\frac{\partial a_i}{\partial y_j} = [F^{-1}]_{ij}$$

Spline Input Quote Sensitivity Jacobian

1. Segment Quote Jacobian: Formulation for quote Jacobian is different than those for the coefficient edge value Jacobian, since the former automatically figures in the design matrix in the sensitivity matrix extractor pseudo-calibration stage. Thus, the quote sensitivities are effectively external sensitivity constraints transmitted via the design matrix quote sensitivities.
2. Quote Jacobian Matrix: The Quote Sensitivity coefficients are calibrated identically to that of the base coefficient sensitivities. This simply follows from the linearity of the quote sensitivity formulation. The only area where there is non-linearity is in the product term $\beta_i \alpha(q_c)$, and that appears only at the constraint equations. Others are identically the same.
3. Latent State Quote Sensitivity: Spline Formulation of the Latent State automatically implies that the quote sensitivity of the latent state is restricted by the above, and is therefore also a spline in itself. This further implies that the boundary formulation is subject to the similar edge conditions as before.
4. Terminology and Nomenclature:
    a. q_c => Input "Calibration Quotes" $q_c \Longrightarrow q_0, \cdots, q_{d-1}$
    b. n_{VM} => Number of explicit Input Node Value Matches
    c. n_{CM} => Number of explicit Input Constraint Value Matches
    d. n_{DM} => Number of explicit Input Derivative Value Matches
    e. n_{PM} => Number of explicit Input Penalty Value Matches
5. Explicit Input-to-Response Match: This emanates from the C^0 node match continuity criterion, i.e., from

$$\sum_{i=0}^{n-1} \beta_i f_i(x_j) = F_j$$

we get that

$$\sum_{i=0}^{n-1} f_i(x_j) \frac{\partial \beta_i}{\partial q_c} = \frac{\partial F_j}{\partial q_c}$$

for

$$j = 0, \cdots, n_{VM} - 1$$

6. <u>Explicit Input-to-Constraint Match</u>: From

$$\sum_{i=0}^{n-1} \beta_i \gamma_{ij} = V_j$$

we can derive that

$$\sum_{i=0}^{n-1} \frac{\partial [\beta_i \gamma_{ij}]}{\partial q_c} = \frac{\partial V_j}{\partial q_c}$$

for

$$j = n_{VM}, \cdots, n_{CM} - 1$$

This may be re-written as

$$\sum_{i=0}^{n-1} \gamma_{ij} \frac{\partial \beta_i}{\partial q_c} = \frac{\partial V_j}{\partial q_c} - \sum_{i=0}^{n-1} \beta_i \frac{\partial \gamma_{ij}}{\partial q_c}$$

7. Explicit Input-to-Derivative Match: This emanates from the C^k continuity criterion

$$\sum_{i=0}^{n-1} \beta_i \left|\frac{\partial f_i(x)}{\partial x}\right|_{x=x_j} = \Lambda_j$$

which results in

$$\sum_{i=0}^{n-1} \frac{\partial \beta_i}{\partial q_c} \left|\frac{\partial f_i(x)}{\partial x}\right|_{x=x_j} = \frac{\partial \Lambda_j}{\partial q_c}$$

for

$$j = n_{CM}, \cdots, n_{DM} - 1$$

8. Explicit Input-to-Penalizer Match: This emanates from the criterion for the curvature and length penalty criterion as

$$\sum_{i=0}^{n-1} \beta_i C_{ij} = P_j$$

which results in

$$\sum_{i=0}^{n-1} \beta_i \frac{\partial C_{ij}}{\partial q_c} = \frac{\partial P_j}{\partial q_c}$$

for

$$j = n_{DM}, \cdots, n_{PM} - 1$$

Shape Preserving Spline

Shape Preserving Tension Spline

1. Integrated vs. Partitioned Shape Controller: Integrated Shape Controllers apply shape control on a basis function-by-basis function basis (certain basis functions such as flat/linear polynomial functions have no explicit shape controller applied on them). Partitioned shape controllers apply shape control on a segment-by-segment basis.
2. Shape Controller Parameter Types:
   - Specified extraneously as part of the basis function formulation itself (e.g., hyperbolic/exponential tension splines)
   - Specified by over-determination of the basis function set (e.g., ν splines)
   - Specified by using a shape controller basis set that is de-coupled from the model basis function set (e.g., partitioned rational splines)
3. Shape Control as part of Basis Function formulation:
   - Each basis function is typically formulated as a linear interpolant of a particular r^{th} derivative across a segment, i.e., $\frac{\partial^r y}{\partial x^r} - \sigma^r y$ is proportional to x^1 in that segment.
   - Advantage is that you can control the switch between the r^{th} derivative and the 0^{th} derivative of y by controlling σ.
   - You can also explicitly formulate it to achieve C^k continuity across segments – and k can vary independently of r.
4. Drawbacks of Shape Control as part of Basis Function formulation:
   - σ may not map well to the curvature/shape departure minimization metrics.
   - The formulation constraint restricts the choice of basis functions, giving rise to possibly unwieldy ones (troubles with exponential/hyperbolic functions are well-documented).
5. Shape Control using over-determined Basis Function Set:
   - Choose any set of basis Functions (e.g., based on simplicity/ease of use/model propriety).
   - Over-specify the set so that additional coefficients are available for explicit and flexible shape control

- Explicit shape control formulation => this comes out a minimization exercise of a "shape departure penalty" function.

6. Drawbacks of Shape Control using Over-determined Basis Function formulation:
   - Ease of use, more model/physics targeted, but comes with extra complexity that trades in flexibility
   - Formulation Complexity => Incorporating variational techniques for enforcing compliance by penalizing shape departure.
   - Functional implementation complexity
   - Jacobian estimation complexity => Now $n \times n$ basis functions for which we need Jacobian.
   - Algorithmic complexity => Need more robust basis inversion/linearization techniques.
7. Potentially Best of Both – Partitioned Basis and Shape Control:
   - Basis function set chosen from physics and other considerations
   - Shape Control achieved using targeted Shape Controllers
   - Used in conjunction with over-determined/other shape control techniques.
8. Drawbacks of Using Partitioned Basis Functions:
   - Choice of shape controllers crucial and non-trivial – they have to satisfy the segment edge and shape variational constraints
   - Need clear and well-specified formulations to match/satisfy the appropriate metrics of shape preservation
   - Formulation Complexity – all calibrations and Jacobians need to incorporate the partitioned basis right during the formulation stage
     - When dealing with snipping/clipping segments, shape controllers operate best at the global level (i.e., at the span/stretch basis). Shape controller smoothness continuity will therefore be ensured, but requires careful formulation at the local/global switch part (i.e., translation of the derivatives using the span/segment scales has to be done carefully).
9. Partitioned vs. Integrated Tension Splines: Partitioned splines are designed such that the interpolant functional and the shape control functional are separated by formulation (e.g., rational splines). Integrated tension splines are formulated such that the shape preservation is

an inherent consequence of the formulation, and there is no separation between the interpolant and the shape control functionality.

- Customization is easier with partitioning on either the control design or the shape preservation dimension.

10. Explicit Shape Preservation Control in Partitioned Splines:

$$y = \frac{\alpha}{\beta}$$

where α is the interpolant, and β is the shape controller. Typically α is determined (among other things) by the continuity criterion C^k, and β contains an explicit design parameter for shape control (for e.g., λ in the case of rational splines).

11. Shape Control Design: Asymptotically, depending on the shape design parameter λ, $\frac{\alpha}{\beta}$ should switch between linear and polynomial (i.e., typically cubic – Qu and Sarfraz (1997)). Further, design β such that

$$\beta_0 = \beta_1 = 1$$

so that

$$y_0 = \alpha_0$$

and

$$y_1 = \alpha_1$$

12. Rational Cubic Spline Formulation: Rational functions under tension was introduced by Spath (1974), and formulation expanded in the general tension setting by Preuss (1976).

$$y = \frac{a + bx + cx^2 + dx^3}{1 + \lambda x(1 - x)} = \frac{\alpha}{\beta}$$

where

$$\alpha = a + bx + cx^2 + dx^3$$

and

$$\beta = 1 + \lambda x(1 - x)$$

(Delbourgo and Gregory (1983), Delbourgo and Gregory (1985a), Delbourgo and Gregory (1985b), Delbourgo (1989)).

$$\lambda \to 0$$

makes it cubic, and

$$\lambda \to \infty$$

makes it linear.

13. Rational Cubic Spline Coefficients:

$$a = 1 \cdot y_0 + 0 \cdot y_1 + 0 \cdot y_0{}' + 0 \cdot y_0{}''$$

$$b = \lambda \cdot y_0 + 0 \cdot y_1 + 1 \cdot y_0{}' + 0 \cdot y_0{}''$$

$$c = -\lambda \cdot y_0 + 0 \cdot y_1 + \lambda \cdot y_0{}' + \frac{1}{2} \cdot y_0{}''$$

$$d = -1 \cdot y_0 + 1 \cdot y_1 + [-(1 + \lambda)] \cdot y_0{}' + \left(-\frac{1}{2}\right) \cdot y_0{}''$$

14. Rational Cubic Spline Derivatives:

$$\alpha = a + bx + cx^2 + dx^3$$

$$\frac{\partial \alpha}{\partial x} = b + 2cx + 3dx^2$$

$$\frac{\partial^2 \alpha}{\partial x^2} = 2c + 6dx$$

$$\beta = 1 + \lambda x(1 - x)$$

$$\frac{\partial \beta}{\partial x} = \lambda(1 - 2x)$$

$$\frac{\partial^2 \beta}{\partial x^2} = -\lambda$$

$$\frac{dy}{dx} = \frac{\beta \frac{\partial \alpha}{\partial x} - \alpha \frac{\partial \beta}{\partial x}}{\beta^2}$$

$$\frac{d^2 y}{dx^2} = \frac{\beta^2 \frac{\partial^2 \alpha}{\partial x^2} - \alpha\beta \frac{\partial^2 \beta}{\partial x^2} + 2\alpha \left(\frac{\partial \beta}{\partial x}\right)^2 - 2\beta \frac{\partial \alpha}{\partial x} \frac{\partial \beta}{\partial x}}{\beta^2}$$

15. Designing λ_i for the Segment Inflection/Extrema Control: If there are "physics" hints, the segment λ_i can be designed to push out/pull in the inflections and/or extrema out of (or into) the segment. For instance, Gregory (1984) and Gregory (1986) established the monotonizing parameters for the rational splines as

$$\lambda_i = \mu_i + [f'(x_i) + f'(x_{i+1})] \frac{x_{i+1} - x_i}{y_{i+1} - y_i}$$

for

$$x_i < x < x_{i+1}$$

$$\mu_{i+1} \geq -3$$

makes it monotone in this segment.

$$\mu_i = -2$$

produces a rational quadratic. Convergence is $\mathcal{O}(h^4)$ in all cases.

16. Co-convex choice for λ_i: A similar analysis can be done to make the spline co-convex, but the corresponding formulation requires a non-linear solution for λ_i.
17. Generalized Shape Controlling Interpolator: Given a pair of points

$$\{x_1, y_1\}, \{x_2, y_2\} \rightarrow \{0, y_1\}, \{1, y_2\}$$

a C^0 spline S_0, and a C^k spline S_k, we define a shape controlling interpolator spline S_C by

$$S_C(x) \propto \frac{1}{[S_k(x) - S_0(x)]^2}$$

with the constraints

$$S_C(x = 0) = S_C(x = 1) = 1$$

Rational Shape Controller described earlier meets these requirements.

18. Generically Partitioned Spline Derivative:

$$y(x) = \alpha(x)\beta(x)$$

$$\frac{\partial y}{\partial x} = \beta \frac{\partial \alpha}{\partial x} + \alpha \frac{\partial \beta}{\partial x}$$

$$\frac{\partial^2 y}{\partial x^2} = \beta \frac{\partial^2 \alpha}{\partial x^2} + 2 \frac{\partial \alpha}{\partial x} \frac{\partial \beta}{\partial x} + \alpha \frac{\partial^2 \beta}{\partial x^2}$$

More generally

$$\frac{\partial^n y}{\partial x^n} = \sum_{r=0}^{n} \binom{n}{r} \frac{\partial^{n-r} \alpha}{\partial x^{n-r}} \frac{\partial^r \beta}{\partial x^r}$$

19. <u>Partitioned Interpolating Spline Coefficient</u>: Given

$$\beta_0 = \beta_1 = 1$$

$$y_0 = \alpha_0$$

$$y_1 = \alpha_1$$

$$\left[\frac{\partial y}{\partial x}\right]_{x=0} = \left[\frac{\partial \alpha}{\partial x}\right]_{x=0} [\beta]_{x=0} + [\alpha]_{x=0} \left[\frac{\partial \beta}{\partial x}\right]_{x=0}$$

becomes

$$\left[\frac{\partial \alpha}{\partial x}\right]_0 = \left[\frac{\partial y}{\partial x}\right]_0 - \alpha_0 \left[\frac{\partial \beta}{\partial x}\right]_0$$

Likewise

$$\left|\frac{\partial^2 \alpha}{\partial x^2}\right|_0 = \left|\frac{\partial^2 y}{\partial x^2}\right|_0 - \alpha_0 \left|\frac{\partial^2 \beta}{\partial x^2}\right|_0 - 2 \left|\frac{\partial \alpha}{\partial x}\right|_0 \left|\frac{\partial \beta}{\partial x}\right|_0$$

Partitioned input micro-Jack for cubic interpolator then may be computed as

$$a = 1 \cdot y_0 + 0 \cdot y_1 + 0 \cdot y_0{}' + 0 \cdot y_0{}''$$

$$b = \left[-\left|\frac{\partial \beta}{\partial x}\right|_0\right] \cdot y_0 + 0 \cdot y_1 + 1 \cdot y_0{}' + 0 \cdot y_0{}''$$

$$c = \left[\left(\left|\frac{\partial \beta}{\partial x}\right|_0\right)^2 - \frac{1}{2}\left|\frac{\partial^2 \beta}{\partial x^2}\right|_0\right] \cdot y_0 + 0 \cdot y_1 + \left[-\left|\frac{\partial \beta}{\partial x}\right|_0\right] \cdot y_0{}' + \frac{1}{2} \cdot y_0{}''$$

$$d = \left[\frac{1}{2}\left|\frac{\partial^2 \beta}{\partial x^2}\right|_0 + \left|\frac{\partial \beta}{\partial x}\right|_0 - \left(\left|\frac{\partial \beta}{\partial x}\right|_0\right)^2 - 1\right] \cdot y_0 + 1 \cdot y_1 + \left[\left|\frac{\partial \beta}{\partial x}\right|_0 - 1\right] \cdot y_0{}' + \left(-\frac{1}{2}\right) \cdot y_0{}''$$

20. <u>Interpolating Polynomial Splines of Degree n</u>: Given

$$y = \sum_{i=0}^{n} \alpha_i x^i$$

$$x \in [0, 1)$$

- Polynomial Basis Series for Representation => Taylor series uses the polynomial basis series for representation, and is popular because of the reasons below (other basis may be more cognitive, and derivative representation using them may be more intuitive as well).
    i. Mathematical simplicity
    ii. Completeness.
- Native link of polynomials to derivatives => Given that derivatives are natively linked polynomial basis function representations, all the lower degree polynomial basis functions (i.e., degree < derivative order) get eliminated, thus only allowing the higher order to survive.
- Polynomial C^k Derivative =>

$$\frac{\partial^r y}{\partial x^r} = \sum_{i=0}^{n} \alpha_i \frac{i!}{(i-r)!} x^{i-r}$$

$$\alpha_0 = y_0$$

$$\alpha_r = \frac{1}{r!}\left[\frac{\partial^r y}{\partial x^r}\right]_{x=0}$$

$$r \in [1, n-1]$$

$$\alpha_n = y_1 - \sum_{r=0}^{n-1}\left\{\frac{1}{r!}\left[\frac{\partial^r y}{\partial x^r}\right]_{x=0}\right\}$$

21. <u>"Derivative Completeness" Nature of Polynomial Basis Function</u>: One big advantage for polynomial basis functions is that they are "derivative complete" in the local as well as global sense, i.e., the $k+1$ basis polynomials are sufficient to uniquely determine the C^k continuity constraint. This is not true of non-polynomial basis functions (exponential basis functions, for e.g., need an infinite number of derivatives for completely derivative coefficient determination), therefore their shape needs global determination.
22. <u>Polynomial Interpolating Spline Coefficient micro-Jack</u>:

$$\frac{\partial \alpha_0}{\partial y_0} = 1; \frac{\partial \alpha_0}{\partial y_1} = 0; \left\{\frac{\partial \alpha_0}{\partial \left[\frac{\partial^r y}{\partial x^r}\right]}\right\}_{\substack{x=0 \\ r \neq 0}} = 0$$

$$\frac{\partial \alpha_k}{\partial y_0} = 1; \frac{\partial \alpha_k}{\partial y_1} = 0; \left\{\frac{\partial \alpha_k}{\partial \left[\frac{\partial^r y}{\partial x^r}\right]}\right\}_{\substack{x=0 \\ r \neq 0}} = \frac{1}{r!}\delta_{rk}$$

$$\frac{\partial \alpha_n}{\partial y_0} = -1; \frac{\partial \alpha_n}{\partial y_1} = 1; \left\{\frac{\partial \alpha_n}{\partial \left[\frac{\partial^r y}{\partial x^r}\right]}\right\}_{\substack{x=0 \\ r \neq 0}} = -\frac{1}{r!}$$

23. Curvature Design in Integrated Tension Splines: Cubic spline is interpolant on $\frac{\partial^2 y}{\partial x^2}$ across the nodes, and linear spline is interpolant on y. Thus, $\frac{\partial^2 y}{\partial x^2} - \sigma^2 y$ (the tension spline interpolant) offers the tightness vs. curvature smoothness trade-off.

- Tightness vs. Smoothness Generalization => $\frac{\partial^k y}{\partial x^k} - \sigma^k y$ is linear in x, given k is even. Of course, for

$$k = 2$$

this describes a tension spline (hyperbolic or exponential). Schweikert (1966) used

$$k = 4$$

to improve the shape preservation characteristics.

24. Basis Function Interpolant:

- $\frac{\partial^2 y}{\partial x^2} - \sigma^2 y$ that is linear in x is satisfiable only by hyperbolic and exponential basis splines.
- $\frac{\partial^4 y}{\partial x^2} - \sigma^4 y$ that is linear in x is satisfiable by hyperbolic, exponential, or sinusoidal basis splines.
- More generally, $\frac{\partial^n y}{\partial x^n} - \sigma^n y$ that is linear in x, and where $n = 4m + 2 \ \forall \ m = 0, 1, \cdots$ is satisfied only by hyperbolic and exponential splines.
- $\frac{\partial^n y}{\partial x^n} - \sigma^n y$ that is linear in x, and where $n = 4m \ \forall \ m = 0, 1, \cdots$ is satisfied only by hyperbolic, exponential, or sinusoidal splines.

25. Integrated Tension Spline Types: Sets containing both exponential and hyperbolic basis splines and a linear spline satisfy $\frac{\partial^2 y}{\partial x^2} - \sigma^2 y$. Exponential basis splines

$$\left\{1, x, e^{\frac{\sigma x}{x_{i+1}-x_i}}, e^{\frac{-\sigma x}{x_{i+1}-x_i}}\right\}$$

and hyperbolic basis splines

$$\left\{1, x, \cosh\left(\frac{\sigma x}{x_{i+1}-x_i}\right), \cosh\left(\frac{-\sigma x}{x_{i+1}-x_i}\right)\right\}$$

are two common examples.

26. Exponential Basis Functions: Using the base segment formulation

$$y = A + Bx + Ce^{\frac{\sigma x}{x_1-x_0}} + De^{\frac{-\sigma x}{x_1-x_0}}$$

or its local recast as

$$y = \alpha + \beta\epsilon + \gamma e^{\sigma\epsilon} + \delta e^{-\sigma\epsilon}$$

we extract the "local" coefficients as

$$\alpha = A + Bx_0$$

$$\beta = B(x_1 - x_0)$$

$$\gamma = Ce^{\frac{\sigma x}{x_1-x_0}}$$

and

$$\delta = De^{\frac{-\sigma x}{x_1-x_0}}$$

and their corresponding local-to-global inverses as

$$D = \delta e^{\frac{\sigma x}{x_1 - x_0}}$$

$$C = \gamma e^{\frac{\sigma x}{x_1 - x_0}}$$

$$B = \frac{\beta}{x_1 - x_0}$$

and

$$A = \alpha - \frac{\beta}{x_1 - x_0}$$

The coefficients may now be calibrated as

$$\alpha = y_0 - \frac{y_0''}{\sigma^2}$$

$$\gamma = \frac{1}{2}\left[\frac{y_0'' + \sigma(y_0' - \beta)}{\sigma^2}\right]$$

$$\delta = \frac{1}{2}\left[\frac{y_0'' - \sigma(y_0' - \beta)}{\sigma^2}\right]$$

and

$$\beta = \frac{\sigma^2(y_1 - \alpha) - y_0 \cosh \sigma - \sigma y_0 \sinh \sigma}{\sigma(\sigma - \sinh \sigma)}$$

The coefficient-to-input sensitivity grid may now be computed from

$$\alpha = 1 \cdot y_0 + 0 \cdot y_1 + 0 \cdot y_0' + \left[\frac{-1}{\sigma^2}\right] \cdot y_0''$$

$$\beta = \left[\frac{-\sigma}{\sigma - \sinh\sigma}\right] \cdot y_0 + \left[\frac{\sigma}{\sigma - \sinh\sigma}\right] \cdot y_1 + \left[\frac{-\sinh\sigma}{\sigma - \sinh\sigma}\right] \cdot y_0{}' + \left[\frac{1 - \cosh\sigma}{\sigma(\sigma - \sinh\sigma)}\right] \cdot y_0{}''$$

$$\gamma = \left[\frac{1}{2(\sigma - \sinh\sigma)}\right] \cdot y_0 + \left[\frac{-1}{2(\sigma - \sinh\sigma)}\right] \cdot y_1 + \left[\frac{1}{2(\sigma - \sinh\sigma)}\right] \cdot y_0{}' + \left[\frac{\sigma - 1 + \cosh\sigma - \sinh\sigma}{2\sigma^2(\sigma - \sinh\sigma)}\right] \cdot y_0{}''$$

$$\delta = \left[\frac{-1}{2(\sigma - \sinh\sigma)}\right] \cdot y_0 + \left[\frac{1}{2(\sigma - \sinh\sigma)}\right] \cdot y_1 + \left[\frac{-1}{2(\sigma - \sinh\sigma)}\right] \cdot y_0{}' + \left[\frac{\sigma + 1 - \cosh\sigma - \sinh\sigma}{2\sigma^2(\sigma - \sinh\sigma)}\right] \cdot y_0{}''$$

The corresponding local ordinate derivatives are

$$\frac{\partial y}{\partial \epsilon} = \beta + \sigma[\gamma e^{\sigma\epsilon} - \delta e^{-\sigma\epsilon}]$$

$$\frac{\partial^2 y}{\partial \epsilon^2} = \sigma^2[\gamma e^{\sigma\epsilon} + \delta e^{-\sigma\epsilon}]$$

$$\frac{\partial^3 y}{\partial \epsilon^3} = \sigma^3[\gamma e^{\sigma\epsilon} - \delta e^{-\sigma\epsilon}]$$

$$\frac{\partial^r y}{\partial \epsilon^r} = \beta\delta_{1r} + \sigma^r[\gamma e^{\sigma\epsilon} + (-1)^r \delta e^{-\sigma\epsilon}]$$

Finally, the first 3 orders of the global-to-local derivative transforms become

$$\frac{\partial y}{\partial x} = \frac{1}{x_1 - x_0} \frac{\partial y}{\partial \epsilon}$$

$$\frac{\partial^2 y}{\partial x^2} = \frac{1}{(x_1 - x_0)^2} \frac{\partial^2 y}{\partial \epsilon^2}$$

$$\frac{\partial^3 y}{\partial x^3} = \frac{1}{(x_1 - x_0)^3} \frac{\partial^3 y}{\partial \epsilon^3}$$

27. <u>Hyperbolic Basis Functions</u>: Here the basis function is

$$y = A + Bx + C \cosh\left(\frac{\sigma x}{x_1 - x_0}\right) + D \sinh\left(\frac{\sigma x}{x_1 - x_0}\right)$$

or its local recast as

$$y = \alpha + \beta\epsilon + \gamma \cosh \sigma\epsilon + \delta \sinh \sigma\epsilon$$

The local to global transformers are

$$x = x_0 + \epsilon(x_1 - x_0)$$

$$\epsilon = \frac{x - x_0}{x_1 - x_0}$$

$$\alpha = A + Bx_0$$

$$\beta = B(x_1 - x_0)$$

$$\gamma = C \cosh\left(\frac{\sigma x}{x_1 - x_0}\right) + D \sinh\left(\frac{\sigma x}{x_1 - x_0}\right)$$

$$\delta = C \sinh\left(\frac{\sigma x}{x_1 - x_0}\right) + D \cosh\left(\frac{\sigma x}{x_1 - x_0}\right)$$

The sensitivity of the grid coefficients are

$$\alpha = 1 \cdot y_0 + 0 \cdot y_1 + 0 \cdot {y_0}' + \left[\frac{-1}{\sigma^2}\right] \cdot {y_0}''$$

$$\beta = \left[\frac{\sigma}{\sigma - \sinh\sigma}\right] \cdot y_0 + \left[\frac{-\sigma}{\sigma - \sinh\sigma}\right] \cdot y_1 + \left[\frac{\sinh\sigma}{\sigma - \sinh\sigma}\right] \cdot {y_0}' + \left[\frac{\cosh\sigma - 1}{\sigma(\sigma - \sinh\sigma)}\right] \cdot {y_0}''$$

$$\gamma = 0 \cdot y_0 + 0 \cdot y_1 + 0 \cdot {y_0}' + \left[\frac{1}{\sigma^2}\right] \cdot {y_0}''$$

$$\delta = \left[\frac{-1}{\sigma - \sinh\sigma}\right] \cdot y_0 + \left[\frac{1}{\sigma - \sinh\sigma}\right] \cdot y_1 + \left[\frac{-1}{\sigma - \sinh\sigma}\right] \cdot {y_0}' + \left[\frac{1 - \cosh\sigma}{\sigma^2(\sigma - \sinh\sigma)}\right] \cdot {y_0}''$$

The corresponding local ordinate derivatives are

$$\frac{\partial y}{\partial \epsilon} = \beta + \sigma[\gamma \sinh(\sigma\epsilon) + \delta \cosh(\sigma\epsilon)]$$

$$\frac{\partial^2 y}{\partial \epsilon^2} = \sigma^2[\gamma \cosh(\sigma\epsilon) + \delta\ \sinh(\sigma\epsilon)]$$

$$\frac{\partial^3 y}{\partial \epsilon^3} = \sigma^3[\gamma \sinh(\sigma\epsilon) + \delta \cosh(\sigma\epsilon)]$$

$$\frac{\partial^r y}{\partial \epsilon^r} = \begin{cases} \sigma^r[\gamma \cosh(\sigma\epsilon) + \delta\ \sinh(\sigma\epsilon)] & r\ is\ even \\ \beta\delta_{1r} + \sigma^r[\gamma \sinh(\sigma\epsilon) + \delta \cosh(\sigma\epsilon)] & r\ is\ odd \end{cases}$$

Finally, the first 3 orders of the global-to-local derivative transforms become

$$\frac{\partial y}{\partial x} = \frac{1}{x_1 - x_0} \frac{\partial y}{\partial \epsilon}$$

$$\frac{\partial^2 y}{\partial x^2} = \frac{1}{(x_1 - x_0)^2} \frac{\partial^2 y}{\partial \epsilon^2}$$

$$\frac{\partial^3 y}{\partial x^3} = \frac{1}{(x_1 - x_0)^3} \frac{\partial^3 y}{\partial \epsilon^3}$$

28. Alternate specifications of the segment interpolation (Trojand (2011)). Renka (1987) provides techniques for setting σ under several circumstances:
    - Finding σ when f is bound.
        - To get the minimum tension factor required we need to find the zeros of f' (Renka (1987)).
    - Finding σ when f' is bound.
        - To get the minimum tension factor required we need to find the zeros of f' (Renka (1987)).
    - Finding σ from the bound values of convexity/concavity (Renka (1987)).
29. Problems with Hyperbolic/Tension Splines:
    - Hyperbolic and exponential functions are time consuming to compute (Preuss (1976)), Lynch (1982)).
    - They are somewhat unstable to wide parameter ranges (Spath (1969), Sapidis, Kaklis, and Loukakis (1988)).
    - In certain cases, reasonable alternatives have been provided by ν splines (Nielson (1974)) and rational splines.

Shape Preserving ν Splines

1. Generic ν Spline Formulation: Approach here is somewhat similar to Foley (1988), although different language/symbology.
    - p-set Basis Splines per each Segment.
    - n Data Points
    - Penalty of degree m
    - C^k Continuity Criterion

- Data Point Set: $\{x_i, y_i\}$
- Spline Objective Function:

$$\Lambda(\hat{\mu}, k, m, n, p, \vec{\lambda}) = \sum_{i=1}^{n-1} \left\{ [Y_i - \hat{\mu}_p(x_i)]^2 + \lambda_i \int_{x_i}^{x_{i+1}} \left[\frac{\partial^m \hat{\mu}_p(x)}{\partial x^m} \right]^2 dx \right\} + [Y_n - \hat{\mu}_p(x_n)]^2$$

2. Number of Unknowns Analysis: In the above

$$p > m$$

and

$$m \le k$$

- Number of equations from the end points per segment => 2.
- Number of equations from the coefficients determined by the C^k Continuity Criterion: k.
- Number of equations from the Shape Optimization Formulation:

$$w \in [0, p - m + 1]$$

- Total number of equations: $k + w + 2$.
- Number of coefficients per segment => $p + 1$.

3. Node matching constraints: Given that we are examining shape preserving splines, on applying the node match criterion

$$Y_i = \hat{\mu}_p(x_i)$$

to $\Lambda(\hat{\mu}, k, m, n, p, \vec{\lambda})$ formulated earlier, we get

$$\Lambda_{NM}(\hat{\mu}, k, m, n, p, \vec{\lambda}) = \sum_{i=1}^{n} \lambda_i \int_{x_i}^{x_{i+1}} \left[\frac{\partial^m \hat{\mu}_p(x)}{\partial x^m}\right]^2 dx$$

where $\Lambda_{NM}(\hat{\mu}, k, m, n, p, \vec{\lambda})$ is the node matched Spline Objective Function.

4. Generic Curvature Optimization Formulation: Using the above, the curvature optimization for spline basis function inside a local segment i corresponds to

$$\Lambda_i = \int_{x_i}^{x_{i+1}} \left[\frac{\partial^m \hat{\mu}_p(x)}{\partial x^m}\right]^2 dx$$

5. Generic Curvature Optimization Minimizer: Given the basis function set

$$\hat{\mu}_i(x) = \sum_{i=0}^{n-1} \alpha_{ik} f_i(x)$$

$$\frac{\partial \Lambda_i}{\partial \alpha_{ik}} = 2\sum_{j=0}^{n-1} \alpha_{ik} \left\{\int_{x_i}^{x_{i+1}} \left[\frac{\partial^m f_j(x)}{\partial x^m}\right]^2 \left[\frac{\partial^m f_k(x)}{\partial x^m}\right]^2 dx\right\}$$

6. Generic Coefficient Constrained Optimization Setup:

$$\frac{\partial \Lambda_i}{\partial \alpha_{ik}} = 0$$

implies

$$\sum_{i=0}^{n-1} \alpha_{ik} Q_{ijk} = 0$$

where

$$Q_{ijk} = \int_{x_i}^{x_{i+1}} \left[\frac{\partial^m f_j(x)}{\partial x^m}\right]^2 \left[\frac{\partial^m f_k(x)}{\partial x^m}\right]^2 dx$$

7. Polynomial Formulation for $\Lambda_{NM}(\hat{\mu}, k, m, n, p, \vec{\lambda})$: For the set of polynomial basis functions, we set

$$\hat{\mu}_P(x) = \hat{\mu}_i(p, x) = \sum_{j=0}^{p} \alpha_{ij} x^j$$

on a segment-by-segment basis.

- We also seek to optimize $\Lambda_{NM}(\hat{\mu}, k, m, n, p, \vec{\lambda})$ on a per-segment basis by re-casting $\Lambda_{NM}(\hat{\mu}, k, m, n, p, \vec{\lambda})$ to

$$\Lambda_i(k, m, p) = \int_{x_i}^{x_{i+1}} \left[\frac{\partial^m \hat{\mu}_i(p, x)}{\partial x^m}\right]^2 dx$$

8. $\frac{\partial^m \hat{\mu}_i(p,x)}{\partial x^m}$:

$$\frac{\partial^m \hat{\mu}_i(p, x)}{\partial x^m} = \frac{\partial^m}{\partial x^m}\left[\sum_{j=0}^{p} \alpha_{ij} x^j\right] = \frac{\partial^m}{\partial x^m}\left[\sum_{j=m}^{p} \alpha_{ij} x^j\right] = \sum_{j=m}^{p} \frac{j!}{(j-m)!} \alpha_{ij} x^{j-m}$$

9. $\Lambda_i(k, m, p)$:

$$\Lambda_i(k,m,p) = \int_{x_i}^{x_{i+1}} \left[\sum_{j=m}^{p} \frac{j!}{(j-m)!} \alpha_{ij} x^{j-m} \right]^2 dx$$

$$= \sum_{j=m}^{p} \sum_{l=m}^{p} \frac{j!}{(j-m)!} \frac{l!}{(l-m)!} \alpha_{ij} \alpha_{il} \left[\frac{{x_{i+1}}^{j-l+2m+1} - {x_i}^{j-l+2m+1}}{j+l-2m+1} \right]$$

10. <u>Minimization of $\Lambda_i(k,m,p)$</u>:

$$\frac{\partial \Lambda_i(k,m,p)}{\partial \alpha_{ij}} = 0$$

results in

$$2\overline{\alpha_{\iota q}} \beta_{qq} + 2 \sum_{\substack{j=m \\ j \neq q}}^{p} \alpha_{ij} \beta_{qj} = 0 \;\; \forall \;\; m \leq j \leq p$$

Here

$$\beta_{qj} = \frac{q!}{(q-m)!} \frac{l!}{(l-m)!} \left[\frac{{x_{i+1}}^{q-l+2m+1} - {x_i}^{q-l+2m+1}}{q+l-2m+1} \right]$$

Thus

$$\overline{\alpha_{\iota q}} = -\frac{1}{\beta_{qq}} \sum_{\substack{j=m \\ j \neq q}}^{p} \alpha_{ij} \beta_{qj}$$

Since

$$\frac{\partial^2 \Lambda_i(k,m,p)}{\partial {\alpha_{ij}}^2} = \beta_{qq}$$

and

$$\beta_{qq} > 0$$

$\overline{\alpha_{\iota q}}$ corresponds to the minimum of $\Lambda_i(k, m, p)$. Thus, if

$$x_i = 0$$

and

$$x_{i+1} = 1$$

β_{qj} becomes

$$\beta_{qj} = \frac{q!}{(q-m)!} \frac{l!}{(l-m)!} \left[\frac{1}{q+l-2m+1}\right]$$

11. Polynomial ν Splines – Number of unknowns:
    - Number of coefficients (unknown) => $p + 1$
    - Number of Nodal Start/End Values (known) => 2
    - Number of Calibrated coefficients from the C^k criterion (known): k
    - Net number of unknowns:

$$p + 1 - 2 - k = p - k - 1$$

12. Ordered Unknown Coefficient Set in Polynomial ν Splines: Given that

$$y_i = \sum_{j=0}^{p} \alpha_{ij} x^j$$

α_{i0} through α_{ik}, as well as α_{ip}, are known. α_{iq} where

$$k + 1 \leq q < p$$

are the unknown coefficients. For e.g., for C^1 cubic polynomial spline, the number of unknowns are

$$p - k - 1 = 3 - 1 - 1 = 1$$

13. Maximum number of equations available from Optimizing ν Splines: Number of equations available from the optimization is

$$p - 1 - m + 1 = p - m$$

- Determinacy criterion => Thus if

  $$p - m < p - k - 1$$

  or

  $$m > k + 1$$

  there are no solutions!
- Alternatively, for completeness, derive m from k as

  $$m = k + 1$$

  for completeness.
- Finally, if

$$k_{Input} < p - 2$$

An optimizing run is needed.

14. Advantage of Basis Curve Optimizing Formulation: This formulation can readily/easily incorporate linearized constraints in an automatic manner – as long as the explicit constraints are re-cast to be specified with the current segment.

Alternate Tension Spline Formulations

1. Kaklis-Pandelis Tension Spline: As described in Kaklis and Pandelis (1990), here

$$f(t) = f(x_i)[1 - t] + f(x_{i+1})t + c_i t[1 - t]^{m_i} + d_i t^{m_i}[1 - t]$$

where

$$t = \frac{x - x_i}{x_{i+1} - x_i}$$

and m_i is the Kaklis-Pandelis shape-controlling tension polynomial exponent.

$$m_i = 2$$

corresponds to the cubic spline interpolant on $[x_i, x_{i+1}]$.

$$m_i \to \infty$$

corresponds to linear interpolant on $[x_i, x_{i+1}]$.

2. Manni's Tension Spline: The methodology is explained in detail in Manni (1996a), Manni and Sablionniere (1997), and Manni and Sampoli (1998). Here

$$f_i(x) = p_i[{q_i}^{-1}(x)]$$

on $[x_i, x_{i+1}]$ where p_i and q_i are cubic polynomials. Further, q_i is strictly increasing in $[x_i, x_{i+1}]$, so that ${q_i}^{-1}$ is well defined (Manni (1996b)).

- The boundary conditions are:

$$f'(x_i) = d_i$$

We further impose that

$$p_i'(x_i) = \lambda_i d_i$$

$$q_i'(x) = \lambda_i$$

$$p_i'(x_{i+1}) = \mu_i d_{i+1}$$

and

$$q_i'(x_{i+1}) = \mu_i$$

(see Manni (2001)). The claim is that if

$$\lambda_i = \mu_i = 1$$

then

$$q_i(x) = x$$

thus f_i becomes cubic. Also if

$$\lambda_i = \mu_i = 0$$

f_i reduces to linear.

3. KLK Splines: Next section is completely devoted to this.

References

- Delbourgo, R., and J. Gregory (1983): C^2 Rational Spline Quadratic Interpolation to Monotonic Data *IMA J. Numerical Anal.* **3** 141-152.
- Delbourgo, R., and J. Gregory (1985a): The Determination of the Derivative Parameters for a Monotonic Rational Quadratic Interpolant *IMA J. Numerical Anal.* **5** 397-406.
- Delbourgo, R., and J. Gregory (1985b): Shape Preserving Piecewise Rational Interpolation *SIAM J. Sci. Stat. Comput.* **6** 967-976.
- Delbourgo, R. (1989): Shape Preserving Interpolation to Convex Data to Rational Functions with Quadratic Numerator and Linear Denominator *IMA J. Numerical Anal.* **9** 123-136.
- Foley, T. (1988): A Shape Preserving Interpolant with Tension Controls *Computer Aided Geometric Design* **5**.
- Gregory, J. (1984): Shape Preserving Rational Spline Interpolation, in: *Rational Approximation and Interpolation* (Graves-Morris, Saff, and Varga (editors)) **Springer-Verlag** 431-441.
- Gregory, J. (1986): Shape Preserving Spline Interpolation *Computer Aided Design* **18** 53-58.
- Kaklis, P. D., and D. G. Pandelis (1990): Convexity Preserving Polynomial Splines of Non-uniform Degree *IMA J. Numerical Anal.* **10** 223-234.
- Lynch, R. (1982): *A Method for Choosing a Tension Factor for a Spline under Tension Interpolation* **M. Sc, University of Texas**, **Austin.**
- Manni, C. (1996a): C^1 Co-monotone Hermite Interpolation via Parametric Cubics *J. Comp. App. Math.* **69** 143-157.
- Manni, C. (1996b): Parametric Shape Preserving Hermite Interpolation by Piecewise Quadratics, in: *Advanced Topics in Multi-variate Approximation* (Fontanella, Jetter, and Laurent (editors)) **World-Scientific** 211-228.

- Manni, C., and P. Sablonniere (1997): Monotone Interpolation of Order 3 by C^2 cubic Splines *IMA J. Numerical Anal.* **17** 305-320.
- Manni, C., and M. L. Sampoli (1998): Co-monotone Parametric Hermite Interpolation, in: *Mathematical Methods for Curves and Surfaces II,* (Daehlen, Lyche, and Schumaker (editors)) **Vanderbilt University Press** 343-350.
- Manni, C. (2001): On Shape Preserving C^2 Hermite Interpolation *BIT* **14** 127-148.
- Nielson, G. (1974): Some Piecewise Polynomial Alternatives to Splines under Tension, in: *Computer Aided Geometric Design* (R. Barnhill, R. Reisenfeld (editors)) **Academic Press** 209-235.
- Pruess, S. (1976): Properties of Splines in Tension *J. Approx. Theory*. **17** 86-96.
- Qu, R., and M. Sarfraz (1997): Efficient Method for Curve Interpolation with Monotonicity Preservation and Shape Control *Neural, Parallel, and Scientific Computations* **5** 275-288.
- Renka, R. (1987): Interpolator Tension Splines with Automatic Selection of Tension Factors **Society for Industrial and Applied Mathematics** *J. ScL. Stat. Comput.* **8 (3)** 393-415.
- Trojand, D. (2011): *Splines Under Tension*.
- Sapidis, N., P Kaklis, and T Loukakis (1988): A Method for Computing the Tension Parameters in Convexity Preserving Spline-in-Tension Interpolation *Numerical Math* **54** 179-192.
- Schweikert, D. G. (1966): An Interpolation Curve Using a Spline in Tension J*ournal of Mathematics and Physics* **45** 312-313.
- Spath, H. (1969): Exponential Spline Interpolation *Computing* **4** 225-233.
- Spath, H. (1974): *Spline Algorithms for Curves and Surfaces* **Utilitas Mathematica Pub. Inc**. Winnipeg.

Koch-Lyche-Kvasov Tension Splines

Introduction

1. Exponential B-Spline Specification: Expounded in detail in Cline (1974), Koch and Lyche (1989), Koch and Lyche (1993), and Kvasov (2000). First extend the knot set with 6 new points t_{-2}, t_{-1}, t_0, t_{M+1}, t_{M+2}, and t_{M+3} such that $t_{-2} < t_{-1} < t_0 < t_1$ and $t_M < t_{M+1} < t_{M+2} < t_{M+3}$, but arbitrary otherwise.
2. Exponential Hat Functions:
    - $B_{j,2}(t) = \psi''(t)$ for $t_j \le t \le t_{j+1}$, and
    - $B_{j,2}(t) = \phi''(t)$ for $t_{j+1} \le t \le t_{j+2}$, where
    - $\psi_j(t) = \dfrac{\sinh[\sigma(t - t_j)] - \sigma(t - t_j)}{\sigma^2 \sinh[\sigma(t_{j+1} - t_j)]}$, and
    - $\phi_j(t) = \dfrac{\sinh[\sigma(t_{j+1} - t)] - \sigma(t_{j+1} - t)}{\sigma^2 \sinh[\sigma(t_{j+1} - t_j)]}$
3. Properties of the $B_{j,k}(t)$ Splines: $B_{j,2}(t)$ as defined above is the basis on top of which all the higher order splines are built. $B_{j,2}(t)$ is non-zero only for $t \in [t_j, t_{j+2}]$, where $j \in [-1, 0, \ldots, M]$.
4. Layout of Base Monic Setup: With reference to figure 9, the monic basis function $B_{j,2}$ may be estimated from the corresponding primitive hat functions ψ and ϕ (referred to as A and B respectively in Figure 9) as:
    - $B_{j,2} = \psi_j''(t)$ for $t_j \le t < t_{j+1}$.
    - $B_{j,2} = \phi_{j+1}''(t)$ for $t_{j+1} \le t < t_{j+2}$.
    - $B_{j,2} = 0$ otherwise.
5. Monic B-Spline Normalizer:

- $C_{j,2}=\int_{t_j}^{t_{j+2}}B_{j,2}(y)dy=\int_{t_j}^{t_{j+1}}\psi_j''(y)dy+\int_{t_{j+1}}^{t_{j+2}}\phi_j''(y)dy$
- $C_{j,2}=\psi_j'(t_{j+1})-\psi_j'(t_j)+\phi_{j+1}'(t_{j+2})-\phi_{j+1}'(t_{j+1})$
- $C_{j+1,2}=\psi_{j+1}'(t_{j+2})-\psi_{j+1}'(t_{j+1})+\phi_{j+2}'(t_{j+3})-\phi_{j+2}'(t_{j+2})$

6. Monic B-Spline Cumulative Normalized Integrand:
   - $\Lambda_{j,2}=0$ for $t<t_j$.
   - $\Lambda_{j,2}=\frac{\int_{t_j}^{t}\psi_j''(t)}{C_{j,2}}$ for $t_j\le t<t_{j+1}$.
   - $\Lambda_{j,2}=\frac{\int_{t_j}^{t_{j+1}}\psi_j''(t)+\int_{t_{j+1}}^{t}\phi_{j+1}''(t)}{C_{j,2}}$ for $t_{j+1}\le t<t_{j+2}$.
   - $\Lambda_{j,2}=1$ for $t\ge t_{j+2}$.
7. Monic B-Spline Scaled Integrand:
   - $\Lambda_{j,2}=0$ for $t<t_j$.
   - $\Lambda_{j,2}=\frac{\psi_j'(t)-\psi_j'(t_j)}{\psi_j'(t_{j+1})-\psi_j'(t_j)+\phi_{j+1}'(t_{j+2})-\phi_{j+1}'(t_{j+1})}$ for $t_j\le t<t_{j+1}$.
   - $\Lambda_{j,2}=\frac{\psi_j'(t_{j+1})-\psi_j'(t_j)+\phi_{j+1}'(t)-\phi_{j+1}'(t_{j+1})}{\psi_j'(t_{j+1})-\psi_j'(t_j)+\phi_{j+1}'(t_{j+2})-\phi_{j+1}'(t_{j+1})}$ for $t_{j+1}\le t<t_{j+2}$.
   - $\Lambda_{j,2}=1$ for $t\ge t_{j+2}$.
8. Monic B-Spline Scaled Integrand: $B_{j,3}(t)=\Lambda_{j,2}(t)-\Lambda_{j+1,2}(t)=\frac{\int_{t_j}^{t}B_{j,2}(y)dy}{\int_{t_j}^{t_{j+2}}B_{j,2}(y)dy}-\frac{\int_{t_{j+1}}^{t}B_{j+1,2}(y)dy}{\int_{t_{j+1}}^{t_{j+3}}B_{j+1,2}(y)dy}$.
9. Quadratic and Cubic Exponential Tension Splines: Higher order splines are recursively defined from $B_{j,k}(t)=\Lambda_{j,k-1}(t)-\Lambda_{j+1,k-1}(t)$ where:
   - $\Lambda_{j,k}(t)=0$ for $t\le t_j$

- $\Lambda_{j,k}(t) = \frac{1}{\Omega_{j,k}} \int_{t_j}^{t} B_{j,k}(y)dy$ for $t_j \le t \le t_{j+k}$
- $\Lambda_{j,k}(t) = 1$ otherwise

Here $\Omega_{j,k} = \int_{t_j}^{t_{j+k}} B_{j,k}(y)dy$. Further, $k = 2$ and $k = 3$ correspond to quadratic and cubic tension splines, respectively.

10. Similarities between Exponential Tension B-Splines and Polynomial B-Splines: Notice the similarities, the iterative higher-order definitions, and the partition of unity as well.
11. Cubic Exponential Tension B-Spline: This corresponds to the $B_{j,k=4}(t)$ case, i.e.,

$g(t) = \sum_{k=j-1}^{j+2} \beta_k B_{k-2,4}(t)$, with validity in the interval $t \in \lfloor t_j, t_{j+4} \rfloor$.

- Explicit Cubic Basis Representation => Using Koch and Lyche (1989), Koch and Lyche (1993), and Kvasov (2000), define:
    - $z_j = \psi_{j-1}(t_j) - \phi_j(t_j)$
    - $z_j' = \psi_{j-1}'(t_j) - \phi_j'(t_j)$, and
    - $y_j = t_j - \frac{z_j}{z_j'}$
    - $b_{j,(1)} = \frac{b_{j+2} - b_{j+1}}{y_{j+2} - y_{j+1}}$
    - $b_{j,(2)} = \frac{b_{j,(1)} - b_{j,(-1)}}{z_{j+1}'}$
- Expanded $g(t)$ in the new basis representation => For $t \in \lfloor t_j, t_{j+4} \rfloor$,

  $g(t) = \sum_{k=j-1}^{j+2} \beta_k B_{k-2,4}(t) = \beta_j + \beta_{j-1,(1)} \{t - y_j\} + \beta_{j-1,(2)} \phi_j(t) + \beta_{j,(2)} \psi_j(t)$. This clearly shows the similarity to the generalized Kaklis-Pandelis tension spline formulation. Retaining $\phi_j(t)$ and $\psi_j(t)$ this way helps generalize to other basis tension splines.
- Expansion of $g(t)$ in basis function terms =>
    - $g(t) = \beta_{j-1}\alpha_{j-1}(t) + \beta_j \alpha_j(t) + \beta_{j+1}\alpha_{j+1}(t) + \beta_{j+2}\alpha_{j+2}(t)$ where

- $\alpha_{j-1}(t) = \dfrac{\phi_j(t)/z_j'}{y_j - y_{j-1}}$
- $\alpha_j(t) = 1 - \dfrac{t - y_j + \phi_j(t)/z_j'}{y_{j+1} - y_j} - \dfrac{\phi_j(t)/z_j'}{y_j - y_{j-1}} + \dfrac{\psi_j(t)/z_j'}{y_{j+1} - y_j}$
- $\alpha_{j+1}(t) = \dfrac{t - y_j + \phi_j(t)/z_j'}{y_{j+1} - y_j} - \dfrac{\psi_j(t)/z_{j+1}'}{y_{j+2} - y_{j-1}} + \dfrac{\psi_j(t)/z_{j+1}'}{y_{j+1} - y_j}$
- $\alpha_{j+2}(t) = \dfrac{\psi_j(t)/z_{j+1}'}{y_{j+2} - y_{j+1}}$

- Robust/Efficient Calculation of Hyperbolics => Renka (1987) and Rentrop (1980) outline some effective methods for this. For small σ, truncated Taylor series is accurate enough.

12. Piecewise Cubic Interpolant Expansion: Remember that, no matter what the basis tension functions are, for piecewise tension C^2 continuity, they are expected to satisfy

$$d_j = \frac{\partial^2 g(t)}{\partial t^2} - \sigma^2 g(t) = \frac{t_{j+1} - t}{t_{j+1} - t_j}\left[\left|\frac{\partial^2 g(t)}{\partial t^2}\right|_{t=t_j} - \sigma^2 g(t_j)\right] + \frac{t - t_j}{t_{j+1} - t_j}\left[\left|\frac{\partial^2 g(t)}{\partial t^2}\right|_{t=t_{j+1}} - \sigma^2 g(t_{j+1})\right] \text{ for}$$

$t \in [t_j, t_{j+1}]$, i.e., this entity varies linearly across the segment.

- This may be re-cast as $d_j = \beta_{j-1}\varpi_{j-1} + \beta_j\varpi_j + \beta_{j+1}\varpi_{j+1}$ where $\varpi_{j-1} = \dfrac{\{1 - \sigma^2\phi_j(t_j)\}/z_j'}{y_j - y_{j-1}}$

$$\varpi_{j+1} = \frac{\{1 - \sigma^2\psi_{j-1}(t_j)\}/z_j'}{y_{j+1} - y_j} \quad \varpi_j = \beta_j\left[1 - \frac{\{1 - \sigma^2\phi_j(t_j)\}/z_j'}{y_j - y_{j-1}} - \frac{\{1 - \sigma^2\psi_{j-1}(t_j)\}/z_j'}{y_{j+1} - y_j}\right]$$

13. Tension Spline Curvature Penalizing Norm: The pure curvature penalizer may now be altered to become a curvature + length penalizer. Thus, $\mathrm{P}_{Curv} = \lambda\int_{t_1}^{t_M}\left[\{y''(t)\}^2 + \sigma^2\{y'(t)\}^2\right]dt$. Notice that both $\{y''(t)\}^2$ and $\sigma^2\{y'(t)\}^2$ (i.e., separated squares) are included individually in the set up.

- Simplification of the curvature penalty norm term for the Tension Splines => $\mathrm{P}_{Curv} = \lambda\int_{t_1}^{t_M}\left[\{y''(t)\}^2 + \sigma^2\{y'(t)\}^2\right]dt = g_{j+1}''g_{j+1}' - g_j''g_j' - d_j[g_{j+1} - g_j]$. A simple proof of this using integration by parts is available in Andersen (2005).

- Penalizing the segment length in addition to the segment curvature is valid for all spline formulations, of course. However, they may not be reducible/simplifiable as much as they are in tension splines.

14. Constrained Optimizer Estimate for λ: If the RMS best-fit error is to be limited to γ^2 where γ^2 is an extraneously specified closeness of fit metric, the constraint may be expressed as $\frac{1}{N}\left[\vec{V}-c\vec{P}\right]^T W^T W\left[\vec{V}-c\vec{P}\right]\leq\gamma^2$.

- Now the optimization minimizer attains contributions from best-fit, curvature penalty, and segment length penalties.
- Step #1: For a given initial guess of λ, find the optimal co-efficient set $\{\beta_i\}_{i=0}^{n-1}$.
- Step #2: Compute $S(\lambda)=[BestFit]^2$. If $S(\lambda)$, you are done. Otherwise use a suitable root finder to extract λ.
- Step #3: The best-fit optimizer precision norm $S(\lambda)$ is a declining function of λ. If a root exists, the root finder procedure should be able to find it.
- λ estimation in the context of curve building is treated in Tanggaard (1997) (using the GCV technique of Craven and Wahba (1979)) and Andersen (2005).

15. Drawbacks of the above method: This involves yet another non-linear root extractor. Other non-linear root extraction parts in curve building are:

- Non-linear boot-strapping
- Non-linear boundary condition in spline calibration.

Thus, the stability of the precision norm technique outlined above is riddled with challenges.

16. Parallel with Hagan-West Forward Interpolator: In Hagan-West (Hagan and West (2006)) minimalist quadratic interpolator, the segment length is incorporated in a slightly different way – as a minimizer of the $k+1$ jumps at the knots (i.e., if C^2, minimizing C^3 at the knots).

17. The other Tension Splines: They all have the property that the tension parameter moves smoothly from cubic (low tension) to linear (high tension), and have different forms for ϕ and ψ. These forms may make them computationally less expensive too.

    - Non-uniform Rational Cubic Tension Spline with linear Denominator =>

- $$\psi_j = \frac{\left(t - t_j\right)^3}{\left(t_{j+1} - t_j\right)\left[1 + \sigma_j\left(t_{j+1} - t\right)\right]\left[6 + 6\sigma_j\left(t_{j+1} - t_j\right) + 2\sigma_j\left(t_{j+1} - t_j\right)^2\right]}$$
- $$\phi_j = \frac{\left(t_{j+1} - t\right)^3}{\left(t_{j+1} - t_j\right)\left[1 + \sigma_j\left(t - t_j\right)\right]\left[6 + 6\sigma_j\left(t_{j+1} - t_j\right) + 2\sigma_j\left(t_{j+1} - t_j\right)^2\right]}$$

- Non-uniform Rational Cubic Tension Spline with Quadratic Denominator =>
  - $$\psi_j = \frac{\left(t - t_j\right)^3}{\left[\left(t_{j+1} - t_j\right) + \sigma_j\left(t - t_j\right)\left(t_{j+1} - t\right)\right]\left[6 + 6\sigma_j\left(t_{j+1} - t_j\right) + 2\sigma_j\left(t_{j+1} - t_j\right)^2\right]}$$
  - $$\phi_j = \frac{\left(t - t_j\right)^3}{\left[\left(t_{j+1} - t_j\right) + \sigma_j\left(t - t_j\right)\left(t_{j+1} - t\right)\right]\left[6 + 6\sigma_j\left(t_{j+1} - t_j\right) + 2\sigma_j\left(t_{j+1} - t_j\right)^2\right]}$$
- Non-uniform Exponential Rational Spline =>
  - $$\psi_j = \frac{\left(t - t_j\right)^3 \exp\left[-\sigma_j\left(t_{j+1} - t\right)\right]}{\left(t_{j+1} - t_j\right)\left[6 + 6\sigma_j\left(t_{j+1} - t_j\right) + 2\sigma_j\left(t_{j+1} - t_j\right)^2\right]}$$
  - $$\phi_j = \frac{\left(t_{j+1} - t\right)^3 \exp\left[-\sigma_j\left(t - t_j\right)\right]}{\left(t_{j+1} - t_j\right)\left[6 + 6\sigma_j\left(t_{j+1} - t_j\right) + 2\sigma_j\left(t_{j+1} - t_j\right)^2\right]}$$

18. Tension Implied by the Basis Function Set: Given the tension C^2 interpolant relation $\frac{\partial f}{\partial x} + \sigma_j^{\,2}\frac{\partial^3 f}{\partial x^3} = Constant$ inside a given segment j, we can infer σ_j from
$$\sigma_j = -\frac{\left.\left|\frac{\partial f}{\partial x}\right|\right|_{x=1} - \left.\left|\frac{\partial f}{\partial x}\right|\right|_{x=0}}{\left.\left|\frac{\partial^3 f}{\partial x^3}\right|\right|_{x=1} - \left.\left|\frac{\partial^3 f}{\partial x^3}\right|\right|_{x=0}}.$$

19. Caveat for using KLK-type Splines for Local State Shape Proxying: Often (and this may be true for other B Splines as a whole too), the B Spline basis choice may produce segment node edge values and their corresponding derivatives of zero. In this case, you may have a singular calibration matrix that does not calibrate. In particular, this is the case for iterated B Splines that are constructed to vanish and fade rapidly at the edges. This poses for problems for segment-local splines (that may span between 0 and 1 within a given segment).
    - How does the KLK formulation avoid this? It is because it is built off of a cubic B Spline, thus works primarily for that case. KLK retains the basis representation out from a workable/calibratable raw cubic B Spline form that is set to be "well-behaved" at the

edges (by definition) at the cubic basis level. The B Splines that follow the typical iterative generation formulation end up "destroying the raw basis construction information" if set up from above (i.e., ψ above the cubic level becomes ψ'' at the lowest hat level).

References

- Andersen, L. (2005): Discount Curve Construction with Tension Splines *SSRN eLibrary*.
- Cline, A. K. (1974): Scalar and Planar Valued Curve Fitting using Splines under Tension *Communications of the ACM* **17** 218-223.
- Craven, P., and G. Wahba (1979): Smoothing Noisy Data with Spline Function: Estimating the Correct Degree of Smoothness by the Method of Generalized Cross Validation *Numerische Mathematik.* **31** 377-403.
- Hagan, P., and G. West (2006): Interpolation Methods for Curve Construction *Applied Mathematical Finance* **13 (2)** 89-129.
- Koch, P. E., and T. Lyche (1989): Exponential B-Splines in Tension, in: *Approximation Theory VI: Proceedings of the Sixth International Symposium on Approximation Theory* **2** C. K. Chui et. al. (editors) **Academic Press** Boston 361-364.
- Koch, P. E., and T. Lyche (1993): Interpolation with Exponential Splines in Tension, in: *Geometric Modeling, Computing/Supplementum* **8** G. Farin et. al. (editors) **Springer Verlag** Wien 173-190.
- Kvasov, B. (2000): *Methods of Shape-Preserving Spline Approximation* **World Scientific Publishing** Singapore.
- Renka, R. (1987): Interpolator Tension Splines with Automatic Selection of Tension Factors **Society for Industrial and Applied Mathematics** *J. ScL. Stat. Comput.* **8 (3)** 393-415.
- Rentrop, P. (1980): An Algorithm for the Computation of Exponential Splines *Numerische Matematik* **35** 81-93.
- Tanggaard, C. (1997): Non-parametric Smoothing of Yield Curves *Review of Quantitative Finance and Accounting* **9** 251-267.

Smoothing Splines

Penalty Minimization Risk Function

1. Penalty Minimizer Estimator Metric: Choice of the "normalized curvature area" shown in figures 5) and 6) are two possible penalty estimator choices. Obviously, closer the area is to zero, the better the penalizing match is.
2. Dimensionless Penalizing Fit Metric: Choosing the representation in 5), and recognizing that the segment is set in the flat base $(0,1)$, we can derive the representation in 7).
3. Dimensionless Penalty Estimator (DPE): Using Figure 7), we now define DPE as

$$DPE = \frac{Area_{ShadedPart}}{Area_{ABCD}} = \frac{\int_{x_i}^{x_{i+1}} \left[\frac{\partial^m \hat{\mu}(x)}{\partial x^m}\right]^2 dx}{\max\left\{\left[\frac{\partial^m \hat{\mu}(x)}{\partial x^m}\right]^2\right\}(x_{i+1} - x_i)}$$

4. Pros/Cons of the above Representation of DPE: If the basis functions have near-delta functional forms (Figure 8), DPE will still remain ≈ 0, and the metric is not very meaningful in that case. Fortunately, such delta-type basis functions are rare.
5. Aggregate DPE Measure: Need a consolidated DPE metric that spans across all the segments in a span, i.e., the span DPE.

Smoothing Splines Setup

1. Process Control using Weights: Dimensionless units (such as Reynolds' number) can effectively account for the ratio of competing natural forces. Similar use can be done for process control to be able to guide/control between 2 or more competing objectives. For example in the instance of the smoothing spline:

- First Objective => Closeness of match using the most faithful reproducer, or curve fit.

- Second Objective => Smoothest curve through the given points, without necessarily fitting them – of course, "smoothest" possible "curve" is a straight line.

2. Penalizing Smootheners: Penalizing smootheners are the consequence of Bayes estimation applied on the Quadratic Penalties with Gaussian Priors (also referred to with maxim "The Penalty is the Prior").
3. Smoothing Spline Formulation: Given $x_1 < x_2 < \ldots < x_n$, and the function μ that fits the points $[x_i, Y_i]$ from $Y_i = \mu(x_i)$ (see Hastie and Tibshirani (1990) and Smoothing Spline (Wiki)). The smoothing spline estimate $\hat{\mu}$ is the minimizer

$$\min \arg\left\{\Lambda\left(\hat{\mu}, \lambda\right) = \frac{1}{n}\sum_{i=1}^{1}\left[Y_i - \hat{\mu}(x_i)\right]^2 + \lambda\int_{x_1}^{x_n}\left[\frac{\partial^k \hat{\mu}(x)}{\partial x^k}\right]dx\right\}$$

- $\Lambda\left(\hat{\mu}, \lambda\right)$ is the **Spline Objective Function**.
- $\frac{1}{n}$ is needed to the left term to make it finite as $n \to \infty$, otherwise λ will also have to be infinite.
- The derivative "k" corresponds to what makes $\left[\frac{\partial^k \hat{\mu}(x)}{\partial x^k}\right]$ linear. Thus, for cubic splines, k = 2.

4. Bias Curvature/Variance Fit Trade-off: Smaller the λ, the more you will fit for bias (low curvature penalty). Bigger the λ, more you fit for curvature/roughness penalty.
5. Curvature Penalty Minimizer Spline: It can be theoretically shown that the curvature penalty minimizer spline is a cubic spline. Here is how.
    - First, notice that any spline of degree >= 0 can reproduce the knot inputs.
    - By default, curvature corresponds to k = 2. Thus, $\left[\frac{\partial^2 \hat{\mu}(x)}{\partial x^2}\right]$ varies linearly inside a segment, thus this becomes the least possible curvature.
    - Higher order splines will have a non-linear curvature.
    - Lesser order (spline order less than 3) will violate the C^2 continuity constraint.

6. Bias Curvature/Variance Fit Trade-off: Smaller the λ, the more you will fit for bias (low curvature penalty). Bigger the λ, more you fit for curvature/roughness penalty.
7. Smoothing Output Criterion:
    - Speed of Fitting
    - Speed of Optimization
    - Boundary Effects
    - Sparse, Computationally Efficient Designs
    - Semi-Parametric Models
    - Non-normal Data
    - Ease of Implementation
    - Parametrically determinable Limits
    - Specialized Limits
    - Variance Alteration/inflation
    - Adaptive Flexibility Possible
    - Adaptive Flexibility Available
    - Compactness of Results
    - Conservation of data distribution moments
    - Easy Standard Errors
8. Smoothing vs. Over-fitting: Since λ is a control parameter, it can always be attained by a parametric specification. To estimate optimal value of λ against over-fitting, use one of the following other additional criteria to penalize the extra parameters used in the fit, such as the following. Each one of them comes with its own advantages/disadvantages.
    - Cross-validation
    - Global Cross-Validation
    - Akaike Information Criterion
    - Bayesian Information Criterion
    - Deviance
    - Kullback-Leibler Divergence metric
9. Segment Stiffness Control: λ may also be customized to behave as a segment stiffener or a penalty/stiffness controller, thus providing extra knobs for the optimization control.

10. Relation of Lagrangian to Smoothing Spline:

- Lagrangian objective function is used to optimize a multi-variate function $L(x, y)$ to incorporate the constraint $g(x, y) = c$ as $\Lambda(x, y, z) = L(x, y) + \lambda[g(x, y) - c]$. Here λ is the Lagrange multiplier.
- Optimized formulation of the smoothing spline is given by minimizing the spline objective function (a form of optimization) $\Lambda\left(\hat{\mu}, \lambda\right) = \frac{1}{n}\sum_{i=1}^{1}\left[Y_i - \hat{\mu}(x_i)\right]^2 + \lambda\int_{x_1}^{x_n}\left[\frac{\partial^k \hat{\mu}(x)}{\partial x^k}\right]dx$. Here λ is the spline objective function shadow price of curvature penalty, i.e., $\frac{\partial\Lambda\left(\hat{\mu}, \lambda\right)}{\partial c} = -\lambda$, where c is the constraint constant defined analogous to the constraint constant in the Lagrangian objective function: $\int_{x_1}^{x_n}\left[\frac{\partial^k \hat{\mu}(x)}{\partial x^k}\right]dx = c$.

Ensemble Averaging vs. Basis Spline Representation

1. Parallel between Spline Representation and Ensemble Averaging Techniques: From a real-valued inference (i.e., regression + transformed classification) point-of-view, there are significant similarities between ensemble averaging techniques and basis spline representation techniques. While ensemble averaging attempts to aggregate over hypotheses set, multi-spline basis representation attempts to represent the splines over the set of the basis spline representation function set. Further, there are also parallels in the way in which the weights are inferred; for basis splines, this is achieved by a combination of best-fit and penalization, whereas for the ensemble aggregator this done via the variance-bias optimization mechanism.
2. Difference: Ensemble averaging may be viewed as a dual pass training exercise effectively – the first training pass trains the individual response basis functions themselves, and the second pass trains the weights across the basis function set.

3. Focus of the Training Passes: Focus of the first pass in ensemble averaging is simply bias reduction, i.e., enhancing the closeness of fit and reduction of Bayes' risk. The second pass performs the ensemble averaging with a view to reducing the variance – this is comparable to the smoothening pass.
4. Ensemble Averaging vs. Multi-Pass State Inference: Boas reduction is comparable to the shape preservation pass, whereas the ensemble averaging/variance reduction is comparable to the smoothening pass.
5. Differences between Ensemble Averaging and Multi-Pass State Inference:
    a. The shape preserving pass ends up emitting a sequence of parameters that correspond to the stretches across the univariate predictor ranges to represent a SINGLE latent state.
    b. The calibration run during the shape preserving pass produces a set of calibrated parameters for the full set of basis splines for shape preservation (i.e., error minimization).
    c. Training each basis function separately during the bias reduction phase of ensemble averaging does not really purport to "infer" any latent state.
    d. The second phase of ensemble aggregation simply re-works the basis weights across all the "low bias" basis functions, again no notion of "re-inferring of states" involved.
    e. The smoothing phase of the multi-phase latent-state construction may work on a latent state quantification metric that is different from the shape preserving pass; while this does loosen the shape preservation impact, it is no more different to the equivalent step in ensemble averaging.

Least Squares Exact Fit + Curvature + Segment Length Penalty Formulation

1. Least Squares Exact Fit vs. Best Fit: Unlike in functional analysis, in machine learning "exact fit" is treated as a rarity in machine learning, as there is presumed to be an irreducible manifest measure generation error. Here we assume that there are processes the result in

"zero manifest measure uncertainties" – in other words, these are "quotes" that are explicitly honored.

2. Nomenclature:
   - $p = 0,...,q-1$ => Points to Fit for the Least Squares Penalty
   - $i = 0,...,n-1$ => The Basis Functions Index
   - $j = 0,...,m-1$ => Number of Ordinate Points
   - r => Curvature Penalizer Derivative
   - s => Length Penalizer Derivative
   - k => k-Separation to be achieved during the Formulative Derivation
3. The Formulation: $\mathrm{P} = \sum_{p=0}^{q-1}\left[\hat{y}_p - y_p\right]^2 + \lambda\int_{x_0}^{x_m}\left\{\left(\frac{\partial^r \hat{y}}{\partial x^r}\right)^2 + \sigma^2\left(\frac{\partial^s \hat{y}}{\partial x^s}\right)^2\right\}dx$ where $\hat{y}(x) = \sum_{i=0}^{n-1}\beta_i f_i(x)$.
4. Segment-Level Decomposition: Segment-level decomposition ensures optimal segment coefficient formulation to within the boundaries of a segment (from the least squares fit point of view) – however, not necessarily global optimum. Further, these optimal constraints provide an extra degree of freedom at the segment level, and not necessarily at the stretch/span level.
5. Segment-Level Decomposition Formulation:

$$\mathrm{P} = \sum_{p=0}^{q-1}\left[\hat{y}_p - y_p\right]^2 + \lambda\int_{x_{j-1}}^{x_j}\left\{\left(\frac{\partial^r \hat{y}}{\partial x^r}\right)^2 + \sigma_j^{\,2}\left(\frac{\partial^s \hat{y}}{\partial x^s}\right)^2\right\}dx = \sum_{p=0}^{q-1}\mathrm{M}_P + \lambda\left[\mathrm{X}_j + \sigma_j^{\,2}\Lambda_j\right]$$ where

$\mathrm{M}_P = \left[\hat{y}_p - y_p\right]^2$, $\mathrm{X}_j = \int_{x_{j-1}}^{x_j}\left(\frac{\partial^r \hat{y}}{\partial x^r}\right)^2 dx$, and $\Lambda_j = \int_{x_{j-1}}^{x_j}\left(\frac{\partial^s \hat{y}}{\partial x^s}\right)^2 dx$.

6. Least Squares Minimization Review: From earlier,
   - $\mathrm{M}_P = \left[\hat{y}_p - y_p\right]^2 = \left[\sum_{i=0}^{n-1}\beta_i f_i(x_p) - y_p\right]^2$.
   - $\frac{\partial \mathrm{M}_P}{\partial \beta_k} = 2 f_k(x_p)\left[\sum_{i=0}^{n-1}\beta_i f_i(x_p) - y_p\right] = 0 \Rightarrow \sum_{i=0}^{n-1}\beta_i f_i(x_p) - y_p = 0$.
   - $\frac{\partial^2 \mathrm{M}_P}{\partial \beta_k^{\,2}} = 2 f_k^{\,2}(x_p) \ge 0$. Therefore a minimum exists.
7. Curvature Penalty Minimization: Again, from earlier,

- $\mathrm{X}_j = \int_{x_{j-1}}^{x_j} \left(\frac{\partial^r \hat{y}}{\partial x^r} \right)^2 dx = \int_{x_{j-1}}^{x_j} \sum_{i=0}^{n-1} \left(\beta_i \frac{\partial^r f_i(x)}{\partial x^r} \right)^2 dx$.
- $\frac{\partial \mathrm{X}_j}{\partial \beta_k} = 2 \sum_{i=0}^{n-1} \beta_i \int_{x_{j-1}}^{x_j} \left(\frac{\partial^r f_k(x)}{\partial x^r} \right) \left(\frac{\partial^r f_i(x)}{\partial x^r} \right) dx = 0 \Rightarrow \sum_{i=0}^{n-1} \beta_i \int_{x_{j-1}}^{x_j} \left(\frac{\partial^r f_k(x)}{\partial x^r} \right) \left(\frac{\partial^r f_i(x)}{\partial x^r} \right) dx = 0$ for each k.
- Further, $\frac{\partial^2 \mathrm{X}_j}{\partial {\beta_k}^2} = 2 \beta_i \int_{x_{j-1}}^{x_j} \left(\frac{\partial^r f_k(x)}{\partial x^r} \right)^2 dx \ge 0$ for $x_j > x_{j-1}$, thus a minimum exists.

8. Segment Length Penalty Minimization: Similar to the curvature penalty, we get,
   - $\Lambda_j = \int_{x_{j-1}}^{x_j} \left(\frac{\partial^s \hat{y}}{\partial x^s} \right)^2 dx = \int_{x_{j-1}}^{x_j} \sum_{i=0}^{n-1} \left(\beta_i \frac{\partial^s f_i(x)}{\partial x^s} \right)^2 dx$.
   - $\frac{\partial \Lambda_j}{\partial \beta_k} = 0 \Rightarrow \sum_{i=0}^{n-1} \beta_i \int_{x_{j-1}}^{x_j} \left(\frac{\partial^s f_k(x)}{\partial x^s} \right) \left(\frac{\partial^s f_i(x)}{\partial x^s} \right) dx = 0$ for each k.
   - Finally, $\frac{\partial^2 \Lambda_j}{\partial {\beta_k}^2} = 2 \beta_i \int_{x_{j-1}}^{x_j} \left(\frac{\partial^s f_k(x)}{\partial x^s} \right)^2 dx \ge 0$ for $x_j > x_{j-1}$, thus a minimum exists.
9. Combining it all:

$$\sum_{i=0}^{n-1} \beta_i \left\{ f_i(x_p) + \lambda \int_{x_{j-1}}^{x_j} \left(\frac{\partial^r f_k(x)}{\partial x^r} \right) \left(\frac{\partial^r f_i(x)}{\partial x^r} \right) dx + \lambda {\sigma_j}^2 \int_{x_{j-1}}^{x_j} \left(\frac{\partial^s f_k(x)}{\partial x^s} \right) \left(\frac{\partial^s f_i(x)}{\partial x^s} \right) dx \right\} = y_p .$$

Alternate Smootheners

1. Compendium of Smoothing Methods:
   - Kernel Smoothing with or without binning.
   - Local Regression with or without binning.
   - Smoothing Splines with or without band solvers.
   - Regression splines with fixed/adaptive knots.
   - Penalizing B Splines.
   - Density Smoothing.

2. Kernel Bandwidth Selector: Kernel bandwidth selection is analogous to the optimal knot point selection employed in the regression spline schemes.
    - Remember that the kernel methods essentially use the periodic functions as their basis functions.
3. Regression Splines: Here the data is simply fit to a (hugely) reduced set of basis spline functions, typically using least squares, without any smoothness penalty.
    - Penalized regression splines are pretty much the same as regression splines in that they do use a reduced set of basis splines. However, they do impose a roughness penalty. Penalized regression splines are also referred to as smoothing splines.
    - Polynomial Regression Splines do curve fitting/regression analysis using selective insertion/removal of knots. Knots are added according to the Rao criterion, and removed according to the Wald criterion.
        - Log Splines are a customization of the polynomial regression splines targeted for density estimation. The log of the density is modeled as a cubic spline.
    - Tension Regression Splines => In addition to the curvature penalty and the least squares fit penalty, tension regression splines also penalize the segment length.
4. Base Density Smoothing Formulation: Log-likelihood density smoothing is analogous to maximizing the multinomial likelihood histogram $\log\left[\prod_{i=1}^{m} p_i^{y_i}\right]$, where y_i is the empirical observation count, and p_i is the probability of finding an observation in the cell i.

References

- Hastie, T. J., and R. J. Tibshirani (1990): *Generalized Additive Models*, **Chapman and Hall**.
- Smoothing Spline (Wiki): *Wikipedia Entry for Smoothing Spline*.

Multi-dimensional Splines

1. 2D Wire Span Surface Stretch: Here a 2D Surface is built off of discrete, separated univariate span chords called wires, and the inner nodes are splined in using a targeted "wiring" spline. While this is naturally less smooth compared to the 2D basis splines (and surface energy minimizing splines) below, it does provide targeted customization of particular chords.
2. Non-symmetrical multi-dimensional Variates: Again, considering 2D as an example, it makes sense to use the basis splines separately across both x_1, x_2, as in $\hat{\mu}(x_1, x_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \beta_{ij} B_i(x_1) B_j(x_2)$.
3. Bi-polynomial 2D Spline: For the 2D Segment Range $x \to [x_i, x_{i+1}]$ and $y \to [y_i, y_{i+1}]$, working in the local variate space $t = \frac{x - x_i}{x_{i+1} - x_i}$ and $u = \frac{y - y_i}{y_{i+1} - y_i}$, we transform the spline basis on to the local representation basis as $f_{i,j}(t,u) = \sum_{k=0}^{m} \sum_{l=0}^{n} \beta_{ij,kl} t^k u^l$.
4. Bi-linear 2D Spline: This produces a C^0 surface. Here $k = l = 1$, therefore the first derivatives (and on) are discontinuous on the grid boundaries. From the observation set $\{z_{i,j}\}$, we get from the boundary match the following values for $\beta_{ij,kl}$:
   - $\beta_{ij,00} = z_{i,j}$
   - $\beta_{ij,01} = -z_{i,j} + z_{i,j+1}$
   - $\beta_{ij,10} = -z_{i,j} + z_{i+1,j}$
   - $\beta_{ij,11} = z_{i,j} + z_{i+1,j+1} - z_{i+1,j} - z_{i,j+1}$
5. Bi-Cubic Interpolation: This produces a C^1 surface. Here $k = l = 3$, therefore the first, second, and the first cross derivatives (i.e., $f(x,y)$, $\frac{\partial f(x,y)}{\partial x}$, $\frac{\partial f(x,y)}{\partial y}$, $\frac{\partial^2 f(x,y)}{\partial x^2}$, $\frac{\partial^2 f(x,y)}{\partial y^2}$, and $\frac{\partial^2 f(x,y)}{\partial x \partial y}$) are continuous across the grid boundaries. From the observation set $\{z_{i,j}\}$, their first derivatives, and their cross derivatives, we get from the boundary match the

following values for $\beta_{ij,kl}$ as before. The common way is to cast these as a sequence of 2D relations by unraveling the continuity constraints, and thereby linearizing the formulation.

6. Symmetrical Multi-dimensional variates: The trivial univariate ordering $x_1 < x_2 < \ldots < x_n$ needs revising in the context of certain multivariates, e.g., symmetrical multivariates (Smith, Price, and Lowser (1974), Graham (1983), and Lee (1989)).
    - A general "distance from focal node" t_i makes to more sense to set in the ascending order. Thus $t_i = \sqrt{(x_i - x_F)^2 + \ldots + (z_i - z_F)^2}$, where $[x_F, \ldots, z_F]$ are the multivariate nodes corresponding to the focal node.
    - Alternatively, the distance from the prior node parametrizer $t_i = \sqrt{(x_{i+1} - x_i)^2 + \ldots + (z_{i+1} - z_i)^2}$ may also work.
    - Use Cartesian/radial/axial basis functions to formulate the segments in terms of the surface vector coefficients in "symmetrical variate" situations.
7. Surface Energy Minimization: Surface energy minimization using the "sigma" tension parameter – formulate equation.
    - Thin Plate Spline (Duchon (1976), Duchon (1977)): This is simply 2D spline interpolation, achieved by minimizing the surface bending energy, the minimization of $E[f] = \int_{R^2} \left|\nabla^2 f(x,y)\right|^2 dxdy$, where $\left|\nabla^2 f(x,y)\right|^2 = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} + \frac{\partial^2 f(x,y)}{\partial x \partial y}$.
    - Thin plate splines are an effective way to achieve surface energy minimization, i.e., for a 2D surface, the smoothing spline surface may be created by the minimization of the following least squares surface spline objective function

$$\Lambda\left(\hat{\mu}, x_1, x_2, \lambda\right) = \frac{1}{n}\sum_{i=1}^{1}\left[Y_i - \hat{\mu}(x_{1i}, x_{2i})\right]^2 + \lambda \int_{x_{11}}^{x_{1n}} \int_{x_{21}}^{x_{2n}} \left[\left(\frac{\partial^2 \hat{\mu}(x)}{\partial {x_1}^2}\right)^2 + \left(\frac{\partial^2 \hat{\mu}(x)}{\partial x_1 \partial x_2}\right) + \left(\frac{\partial^2 \hat{\mu}(x)}{\partial {x_2}^2}\right)^2\right] dx_1 dx_2$$

    . Again, apparently this is more appropriate if x_1, x_2 are symmetrical.
8. Elastic Maps Method for Manifold Learning: This method combines the least squared penalty for the approximation error with the bending/torsional-stretching penalty for the proxy manifold. It then uses a coarse discretization to extract the solution for the optimization problem.

References

- Duchon, J. (1976): Interpolation des Fonctions de Deux Variables suivant le Principe de a Minces *RAIRO Analyse Numerique* **10** 5-12.
- Duchon, J. (1977): Splines minimizing Rotation-Invariant Semi-norms in Sobolev Spaces *Lecture Notes in Mathematics (ZAMP)* **57** 85-100.
- Graham, N. Y. (1983): *Smoothing with Periodic Cubic Splines*.
- Lee, E. T. Y. (1989): *Choosing Nodes in Parametric Curve Interpolation*.
- Smith Jr., R. E., J. M. Price, and L. M. Howser (1974): *A Smoothing Algorithm Using Cubic Spline Functions*.

Figure #1
SEGMENT/SPAN Structure Layout

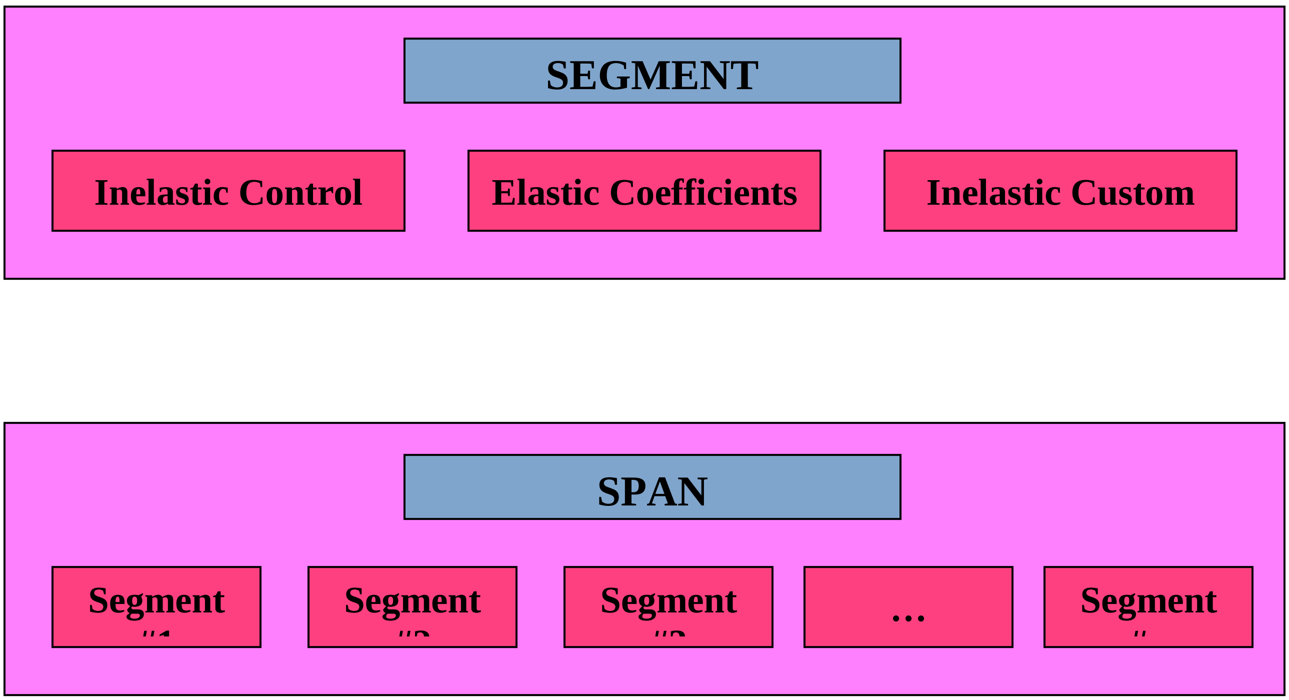

Figure #2
BASIS SPLINE HIERARCHY

SEGMENT

Basis Spline Polynomial	**Basis Spline Hyperbolic Tension**
Basis Spline Linear Polynomial	**Basis Spline Exponential Tension**
Basis Cubic Polynomial	**Basis Spline Cubic Hermite**
Basis Quartic Polynomial	**Basis Spline Catmull Rom**
Basis Cubic Rational	**Basis Spline Cubic Cardinal**

Segment

Segment

Figure #3
THIRD ORDER SECOND DEGREE SPLINE

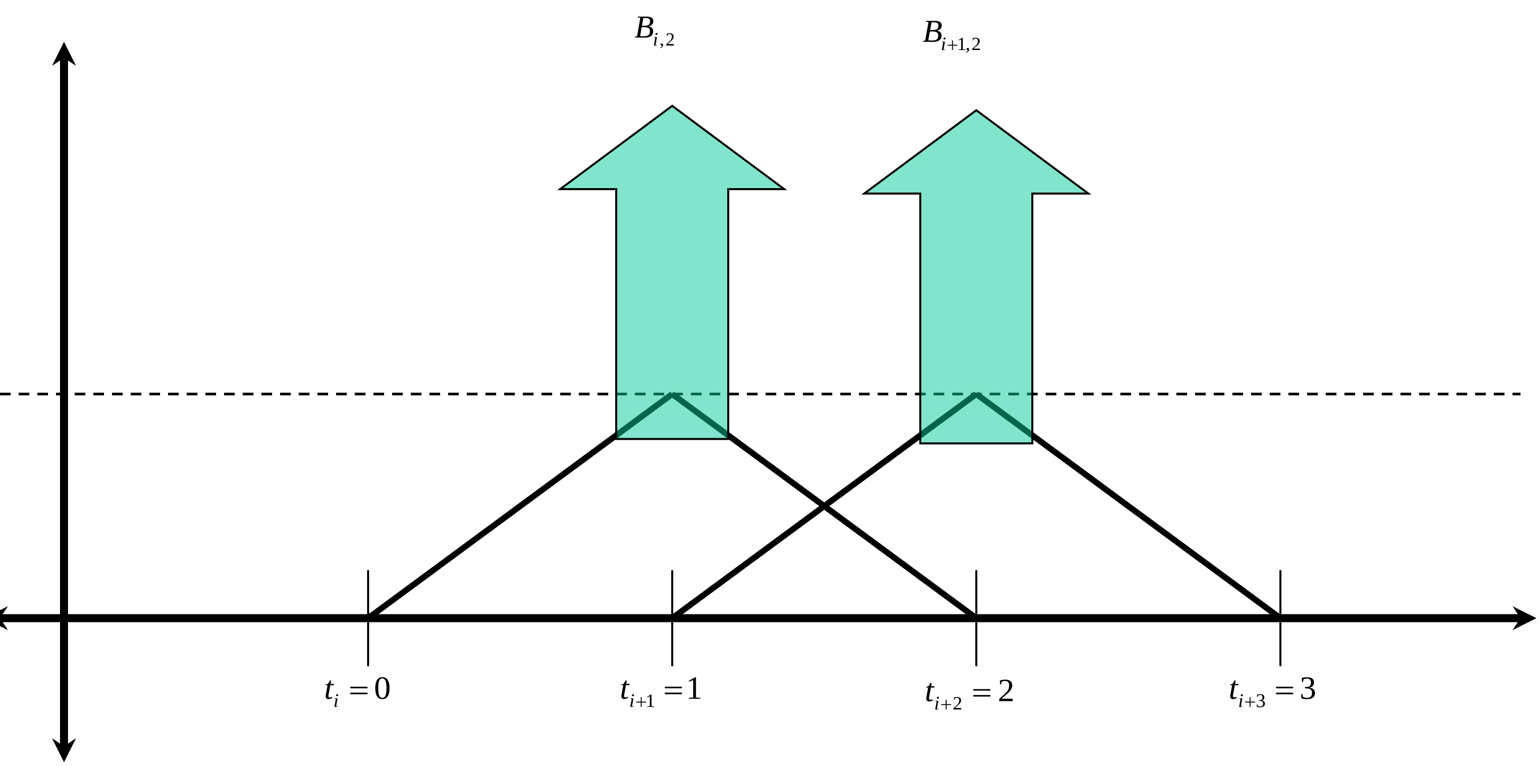

Figure #4
B SPLINE INTERPLATION SCHEME

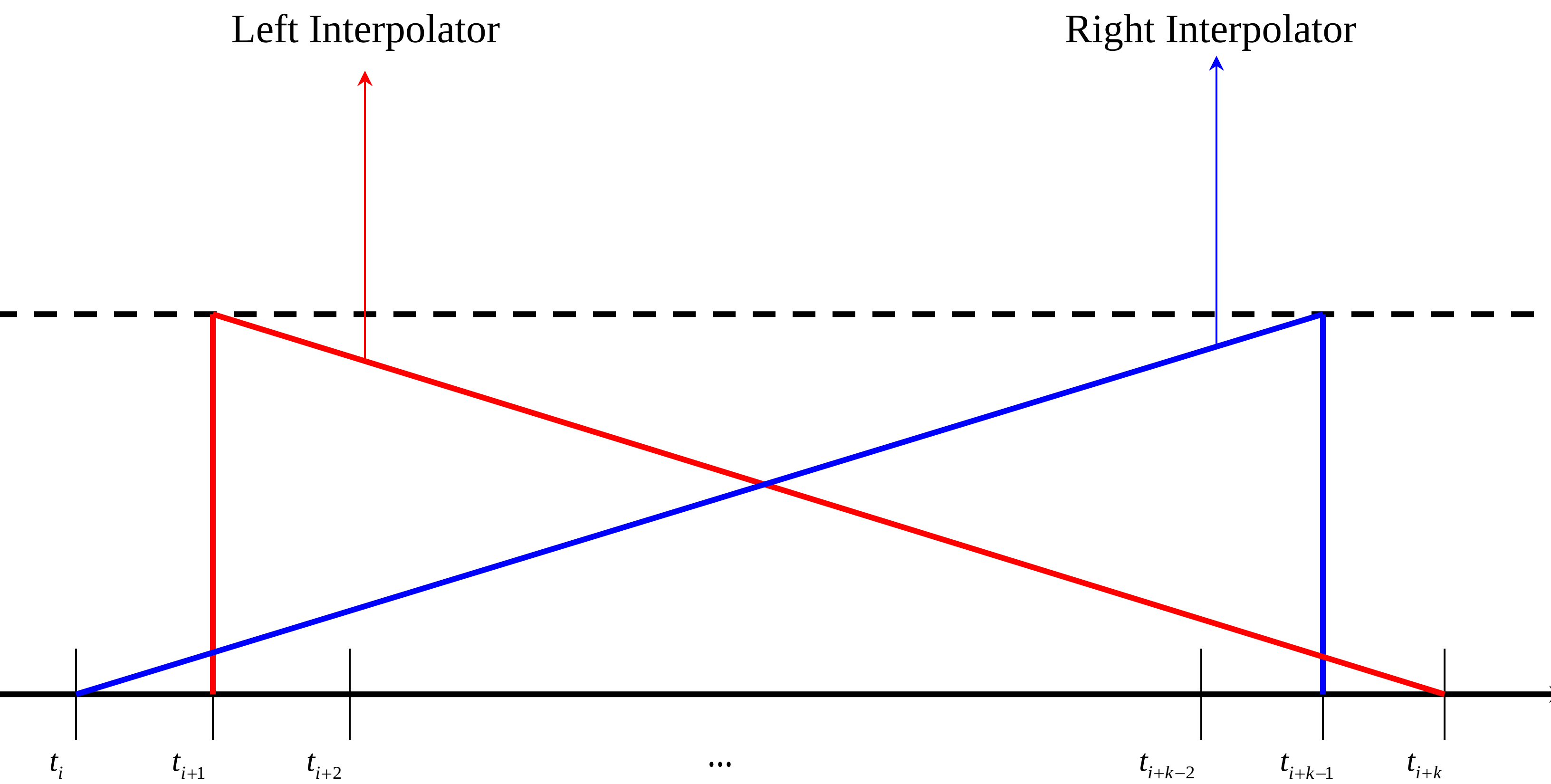

Figure #5
PENALTY MINIMIZER METRIC - #1

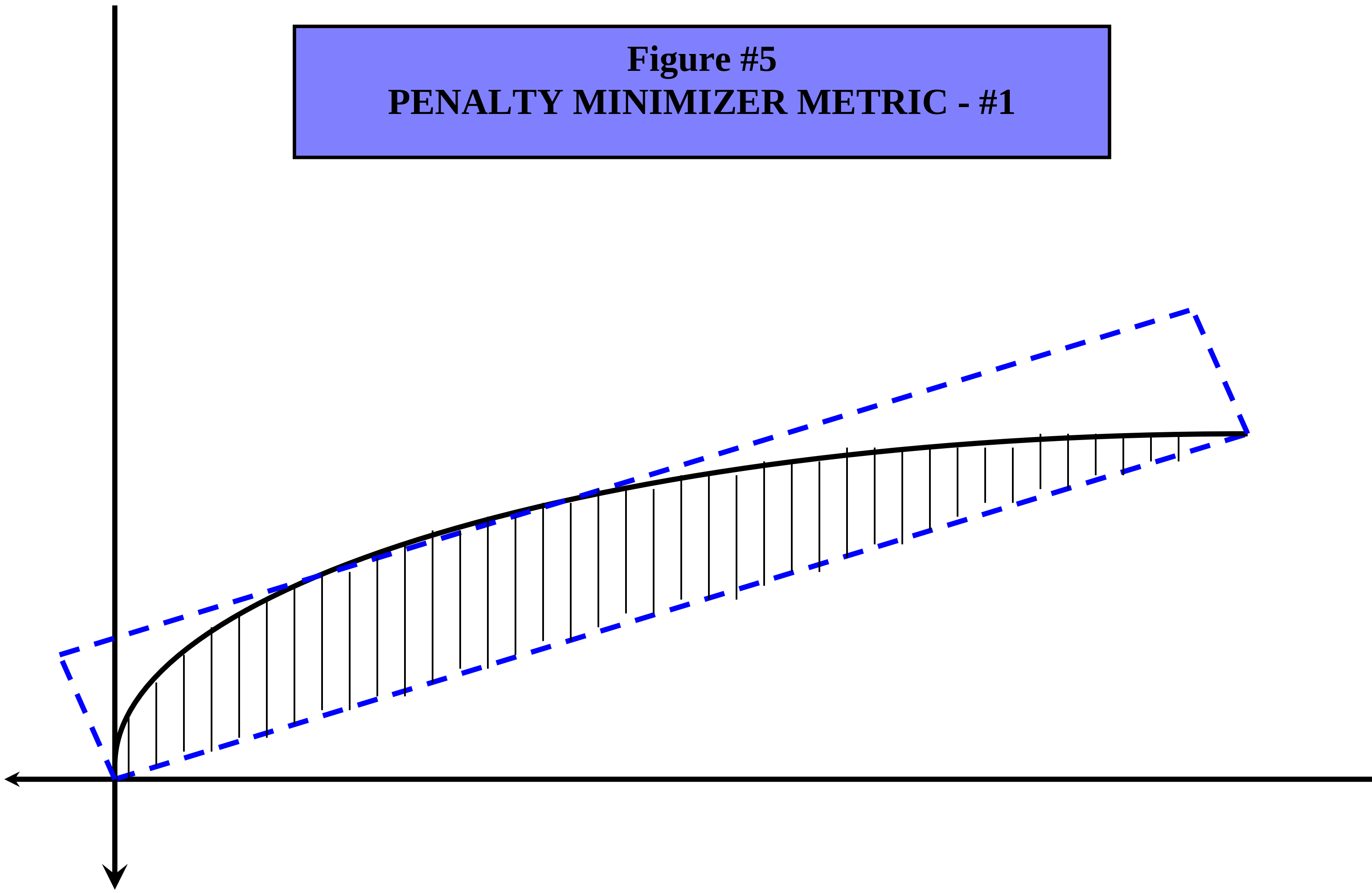

Figure #6
PENALTY MINIMIZER METRIC - #2

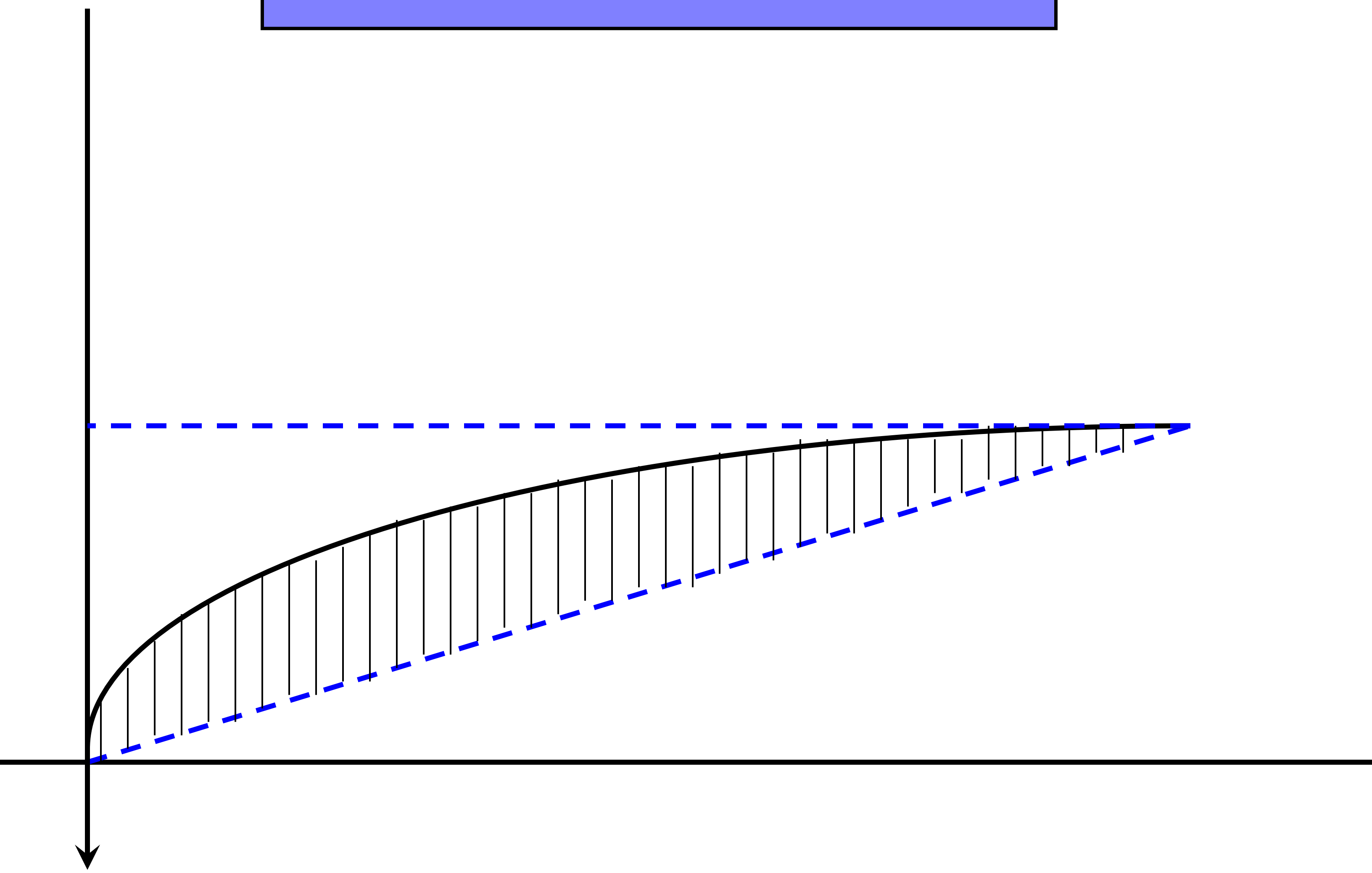

Figure #7
DIMENSION-LESS PENALTY ESTIMATOR

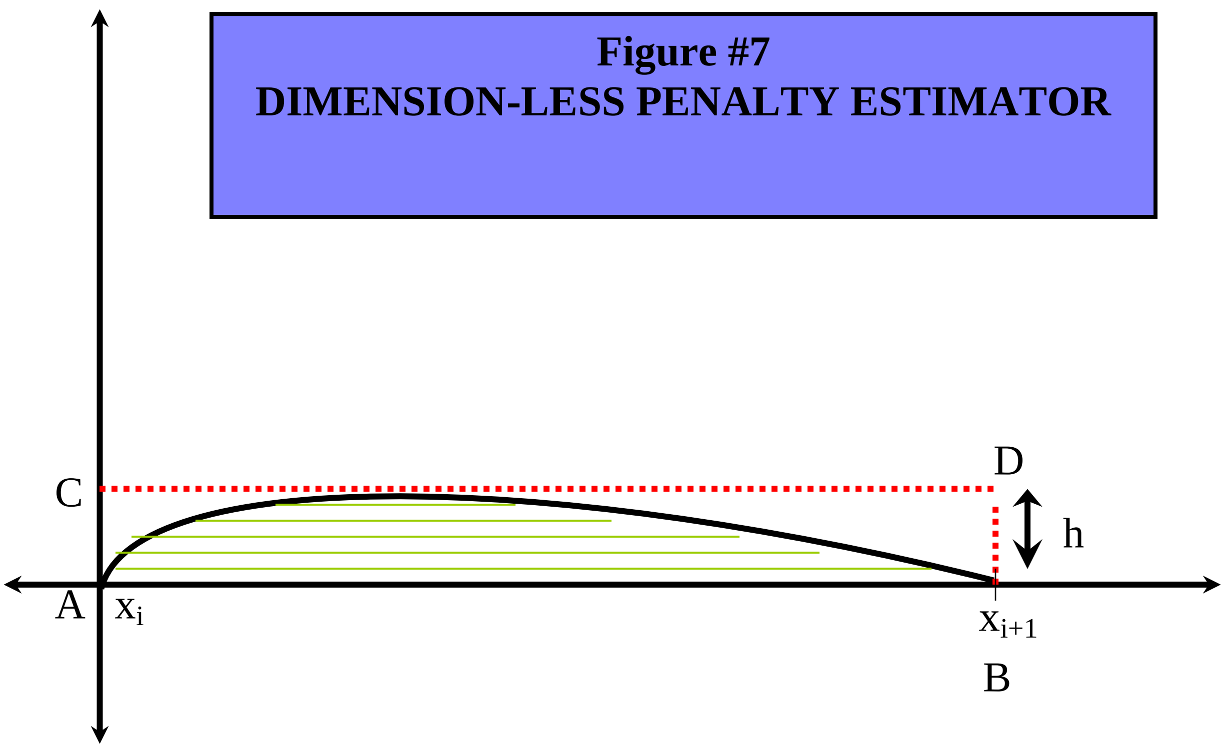

Figure #8
NEAR-DELTA DPE

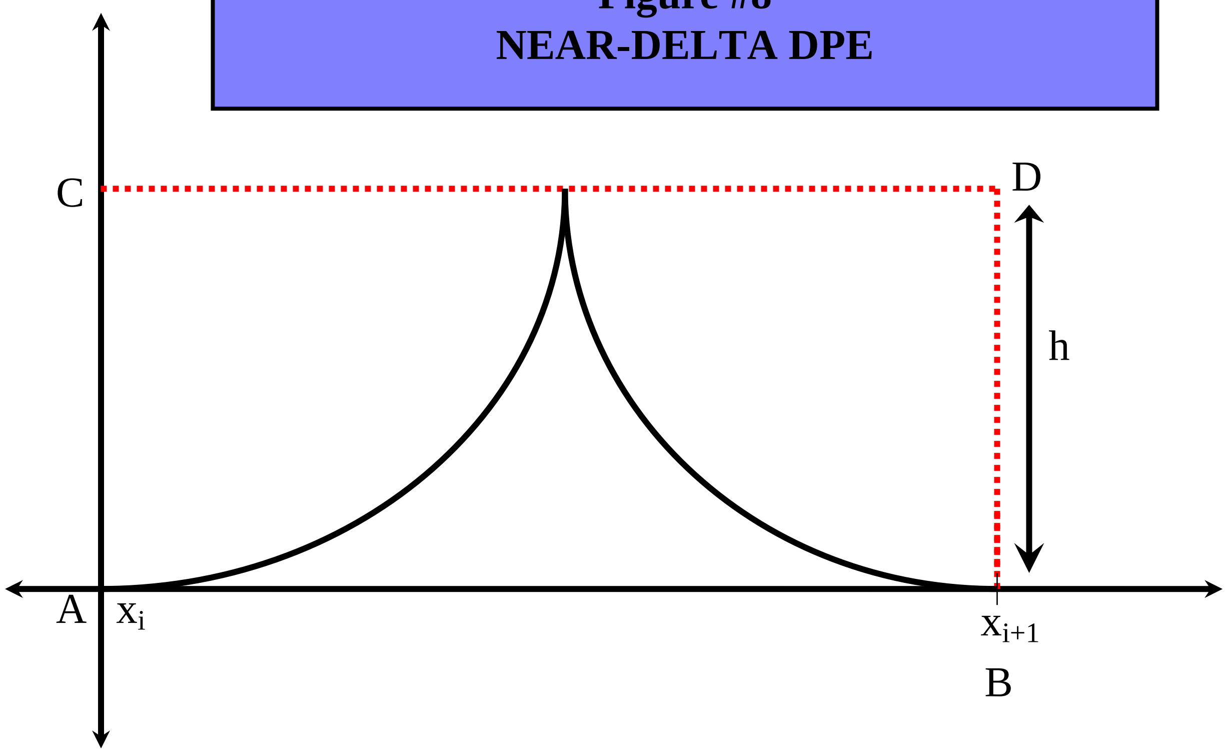

Figure #9
Monic B Spline Base Setup