Spline Library

Lakshmi Krishnamurthy

v0.35, 9 August 2013

Introduction

Framework Symbology and Terminology

1. Response Values: The nodal segment calibration values are also referred to as response values.
2. 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.
3. 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. This automatically renders the coefficient matrix banded (often tri-diagonal)
4. Bias: This is the left hand term in the Spline Objective Function – essentially measures the exactness of fit.
5. Variance: This is the right hand term in the Spline Objective Function – essentially measures the curvature/roughness.

Motivation, Advantage and Purpose

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. Advantages:
    a. Lower degree, gets rid of oscillation associated with the higher degrees [Runge's phenomenon (Wiki)]
    b. Easy, accurate higher degree smoothness specification

3. Applications:
    - Polynomial interpolation
    - Function approximation
    - Surface/contour representation
    - Computer graphics
    - CAD
    - Functional Solution Proxy (Differential Equations, Sensitivity Jacobians etc)
    - Segment/span Calibration
    - Model Inference Extraction
    - Functional Basis Decomposition
    - Image/Signal processing
    - Finance (e.g., Curve Construction)

Literature Review

1. Basic Spline: Covered in [Spline (Wiki), Bartels, Beatty, and Barsky (1987), Judd (1998), Fan and Yao (2005), Chen (2009), Katz (2011)].
2. History: Schoenberg (1946), Ferguson (1964), Epperson (1998).

Purpose of the Document

1. Spline design/SKU objectives, state-of-the art survey.
2. Calibration SKU establishment, and techniques
3. Spline type categorizations
4. Design objective match criteria establishment
5. Mathematical Local/global formulation
6. Jacobian
7. Span/Segment Control Parameters

8. Multi-dimensional Splines
9. Smoothing Spline / Variational Techniques
10. Surface Construction / Fitting / Image / Contour Representation
11. Spline Analytics software SKU partitioning and construction
12. API Usage and Samples discussion

Spline Builder SKU

Design Objectives behind Interpolating Splines

1. Symbols and Definition:
   - Good overview of the desired characteristics is provided in Goodman (2002).
   - Data: $\{x_i, y_i\} \in R^2, i = 0,..., N; x_0 < x_1 < ... < x_i < ... < x_N$
   - Interpolating function: $f(x_i) = y_i; f : [x_0, x_N] \rightarrow [R^2 \Rightarrow R]$
   - Optional Actual Function: $g(x)$
2. **Monotonicity**: $f(x_i)$ 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 extremum is allowed in $\{x_i, x_{+1i}\}$.
3. **Convexity**: $f(x_i)$ 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 $\{x_i, x_{+1i}\}$.
4. **Smoothness**: **Smoothness** (also called **shape-preserving**) corresponds to the least curvature. Even C^0 can be "**smooth**", and so is C^k.

5. **Locality**: **Locality** means that the dependence of $f(x)$ is primarily only on $f(x_i)$ 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 at which $f(x)$ departs from $g(x)$ as the density of x increases. More formally, it is the m in $\|f - g\| \approx O(h^m)$, where $h = \max\{x_{i+1} - x_i : i = 0,..., N-1\}$.
    - For spline segments where $g(x)$ through $g(x)$ 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.

Spline Calibration Framework

1. Definition: Calibration is the process specifying "mandatory" and "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. Two types of static fields: Elastic and inelastic
    - Elastic Fields => Unconstrained change of elastic fields do not force adjustment response (i.e., force disequilibrium).

    - Typically elastic fields correspond to more volatile properties (e.g., temperature of a solid body, quotes of the instruments etc.)
- Inelastic Fields => Unconstrained change of inelastic field forces re-adjustment response (i.e., disequilibrium, followed by stabilization)
    - 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.

3. Calibrator Creation: On creation, objects acquire specific values for the constitutive inelastic fields. Volatile elastic fields may as yet be undefined.
    - Setting of the elastic fields => 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 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 calibration exercise) is inference too.
5. Calibration and entity-variate focus:
    - De-convolving the instrument entity/measure combination is necessary for the extraction of the parameter set (this is accomplished by the calibration process).
    - Of course, calibration occurs among the elastic and the inelastic dimensions, and the inelastics are parameter set!
    - Parameter calibration/parameterization etc: inherently involve parsimonization – this is where the models come in.
6. Curve 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. Calibration Boundary Condition: If the calibration metric is based off of a derivative whose degree is greater than that used to compose the interpolated segments, then the metric will become discontinuous, and thus calibration will fail. For e.g., if you impose continuity only across the first derivatives of all the segments, then a calibration metric that depends on the 2^{nd} derivative (e.g., financial boundary conditions) will fail.
8. Directionality Bias: Directionality "bias" is inherent in calibration (e.g., left to right, ordered sequence set, etc:) – as noted earlier, this simplifies the solution space significantly. Therefore, the same directional bias also exists in the calibration nodal sequence.
9. Head Node Calibration: Calibration of the head 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 matrix parlance, as long the transition 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 ever so periodically (for a full re-build, as opposed to the increments).
- Remember that AD based parametric updates break smoothness (including continuity as Bayesian MLE's) – so use a tolerance in the shift if this is acceptable.

12. Span/Segment Elastic Variates: There are 5 different kinds.
    - Φ => Span stochastic evolution variate.
    - Φ_k => Stochastic evolution variate for segment k.
    - ϕ => Implied Span Quoted Instrument Measure.
    - ϕ_k => Implied Quoted Instrument Measure for Segment k.
    - φ_k => Observed Quoted Instrument Measure for Segment k at precisely a single variate point – typically, the observations are done at the anterior/posterior terminal ends of the segment.
13. Spline Segment Calibrator: Spline segment calibration has an asymmetrical dependence on the left/right calibration value. For a given span, the calibration of the non-left most segment depends only on the right most value – the other coefficients come from the prior segments. The left most segment, of course, uses both the left/right values for calibration.

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:

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

2. Linear 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 (independent of the response values themselves, or their nodal derivative inputs).
3. Analogies to energy minimization over stretches/surfaces
   - Include notes on segment naturalization
4. Span Boundary Specification:
   - "Natural" Spline – Energy minimization problem
   - "Financial" Spline
5. Right Segment Locality Reduction: As you go from the left segment to the right segment, the local segment perturbation impact diminishes due to the fact that information gets transmitted (through the C^1, and C^2 continuity constraints) to the right from the left. Locality is enhanced if, in some sense, "local" information > the transmitted information. Interpolating splines strive to achieve this.
6. 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.
7. Segment Elastics: These are effectively the same as shape controller, i.e., the following are the shape controlling elastic parameter set:
   - Tension σ
   - Number of basis n
   - Continuity C^k

- Optimizing derivative set order m

B-Splines

1. B-Spline Context Fixation: As postulated by de Boor et. al., B Splines have a geometric interpolant context – thereby with the correspondingly strong CADG/curve/surface construction focus. Geometric 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 facilitates surface construction, although there are a lot of similarities with shape-preserving interpolation splines.
2. Lagrange Polynomial vs. kth Order B-Spline Interpolant: Higher order B-Splines are defined by the recurrence $B_{i,k} = \varepsilon_{i,k} B_{i,k-1} + (1 - \varepsilon_{i+1,k}) B_{i+1,k-1}$ where $\varepsilon_{i,k} = \frac{t - t_i}{t_{i+k-1} - t_i}$ and $B_{i1}(t) = X_i(t) = 1$ if $t_i < t < t_{i+1}$, and $B_{i1}(t) = X_i(t) = 0$ otherwise.
   - Coinciding knots => $t_i = t_{i+1} \rightarrow B_{i1} = 0$.
3. 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}.
   - The following provides the insight behind the B Spline interpolation formulation. $B_{p,q}$ spans all the segments between the nodes $[p...q]$. Thus, it is natural to have the interpolator span that segment too.
   - 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.

4. 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 of C^{n-2}.
5. Expository Formulation:
   - $B_{i,k} = \sum_{j=0}^{k} \alpha_{ij} X_{i+j}$
   - $\alpha_{i0} = \varepsilon_{i,k-1} ... \varepsilon_{i0} = \prod_{j=k-1}^{0} \varepsilon_{i,j}$ 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 $\varepsilon_{i+l-1} = \frac{t - t_{i+l-1}}{t_{i+l} - t_{i+l-1}}$
6. Spline Coefficient Partition of Unity: Using the earlier formulation $B_{i,k} = \sum_{j=0}^{k} \alpha_{ij} X_{i+j}$, it is easy to show that $\sum_{j=0}^{k} \alpha_{ij} = 1$. This simply follows from the recursive nodal interpolation property.
7. Smoothness Multiplicity Order Linker: # smoothness conditions at knot + the multiplicity at the knot = B-Spline Order.
8. 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 $\varepsilon_{i,j} = \frac{t - t_{i+j-1}}{t_{i+j} - t_{i+j-1}}$ - $t_{i+j} - t_{i+j-1}$ - increases in length.
9. Other Single B-Spline Properties:
   - B_{ik} is a piece-wise polynomial of degree $< k$ ($k-1$ if the knots are distinct, lesser if the some of the knots coincide).
   - B_{ik} is zero outside of $[t_i, t_{i+k})$.
   - B_{ik} is positive in the open interval $[t_i, ..., t_{i+k}]$.

10. 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

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

- Nodal B-Spline Recursion Stepper:

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

- Spline Order B Spline Recursion Stepper:

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

11. Cardinal B-Spline Knot Sequence: Knot sequence Z => Uniformly spaced knots, simplifying the interpolant/recursive analysis significantly - $\mathrm{Z} \Rightarrow \{..., -2, -1, 0, 1, 2, ...\}$.

- 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
$1 \le t < 2$	$\frac{1}{2}\left(-2t^2 + 6t - 3\right)$	$-2t-3$

$2 \le t < 3$	$\frac{1}{2}(t-3)^2$	$t-3$

12. Non-coinciding B Spline Segment Relations:
    - $B_{i,1} = X_i = 1$ if $t_i \le t < t_{i+1}$
    - $B_{i,1} = X_i = 0$ outside
    - $B_{i,2} = \left[\frac{t-t_i}{t_{i+1}-t_i}\right]X_0 + \left[\frac{t_{i+2}-t}{t_{i+2}-t_{i+1}}\right]X_1$
    - $B_{i+1,2} = \left[\frac{t-t_{i+1}}{t_{i+2}-t_{i+1}}\right]X_1 + \left[\frac{t_{i+3}-t}{t_{i+3}-t_{i+2}}\right]X_2$
    - $B_{i,3} = \left[\frac{t-t_i}{t_{i+2}-t_i}\right]B_{i,2} + \left[\frac{t_{i+3}-t}{t_{i+3}-t_{i+i}}\right]B_{i+1,2}$

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

13. Bernstein B-Spline Knot Sequence: Knot sequence $\Xi \Rightarrow \{0,...,0,1,...,1\}$ - $\{0,...,0\}$ occurs μ times, and $\{1,...,1\}$ occurs ν times.
    - $B[\mu,\nu,t] = \frac{(\mu+\nu)!}{\mu!\nu!}(1-t)^\mu t^\nu$ for $0 \le t < 1$.
    - $B[\mu,\nu,t]$ has $\nu - 1$ derivatives at $t = 0$, and $\mu - 1$ derivatives at $t = 1$ - this is also referred to as ν **smoothness conditions** at $t = 0$, and μ **smoothness conditions** at $t = 1$.
14. B-Spline vs. Spline: B-Spline is just a single polynomial that is valid across a set of knots. "**Spline**" is a linear combination of such B Splines – i.e., the set of all the B_{ik}'s.

15. Spline Definition: $S_{k,t} = \sum_i B_{ik} a_i$ where $a_i \in R^1$. 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}	$\frac{t_{i+3} - t_i}{t_{i+2} - t_{i+1}}$	$\frac{t_{i+3} - t_i}{t_{i+2} - t_{i+1}}$
t_{i+3}	0	-

3. B Spline Continuity Condition: 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.

Local Interpolating Splines

1. Hermite Cubic Splines: The "local information" here takes the form of user specified left/right slopes.
    a. 2 User Specified local slopes + 2 points => 4 sets of equations. Solve for the coefficients.
    b. C^1 continuity is maintained, and C^2 continuity is not.
    c. Segment control is completely local.
2. Catmull-Rom Cubic Splines: Instead of explicitly specifying the left/right segment slopes, they are inferred from the "averages" of the prior and the subsequent points, i.e., $\vec{\tau}_i = \frac{1}{2}\left[\vec{p}_{i+1} - \vec{p}_{i-1} \right]$, and $\vec{\tau}_{i+1} = \frac{1}{2}\left[\vec{p}_{i+2} - \vec{p}_i \right]$. Here $\vec{\tau}_i$ refers to the slope vector, and $\vec{p}_i$ to the point vector.
    a. Again, C^1 continuity is maintained, and C^2 continuity is not.
    b. Segment control is not completely local, but still local enough – it only depends on the neighborhood of 3 points.
3. Cardinal Cubic Splines: This is a generalization of the Catmull-Rom spline with a tightener coefficient σ, i.e., $\vec{\tau}_i = \frac{1}{2}(1-\sigma)\left[\vec{p}_{i+1} - \vec{p}_{i-1} \right]$, and $\vec{\tau}_{i+1} = \frac{1}{2}(1-\sigma)\left[\vec{p}_{i+2} - \vec{p}_i \right]$.

    $\sigma > 0$ corresponds to tightening, and $\sigma < 0$ corresponds to loosening.
    a. Again, C^1 continuity is maintained, and C^2 continuity is not.
    b. Segment control is "local" in the Catmull-Rom sense - it only depends on the neighborhood of 3 points.

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., $x_1 < x_2 < ... < x_n$.

- If the ordering $x_1 < x_2 < ... < x_n$ 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 $u_1 < u_2 < ... < u_n$. The corresponding control points are $[u_1, x_1]..[u_n, x_n]$ and $[u_1, y_1]..[u_n, y_n]$.
    a. Side effect of this – is that you need to work on two pairs 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.

Spline Calibration

1. Spline Segment Calibrator: Spline segment calibration has an asymmetrical dependence on the left/right calibration value. For a given span, the calibration of the non-left most segment depends only on the right most value – the other coefficients come from the prior segments. The left most segment, of course, uses both the left/right values for calibration.
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. Span-to-segment constraint transmission
4. Number of unknowns analysis

Spline Jacobian

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 $\frac{\partial y(x)}{\partial y(x_i)}$.
    - Self-Jacobian tells you the story of sensitivity/perturbability of the interpolant (y) around non-local points. Among the splines, quadratic and greater splines cause fairly non-banded, dispersed Jacobians, indicating that the impact is non-local; linear splines produce simple banded/tri-diagonal Jacobians; and tension splines produce a combination of the two depending on the tension parameter.
    - Further Jacobian of any function $F(Y)$ is going to be dependent on the self-Jacobian $\frac{\partial Y(t)}{\partial Y(t_K)}$ because of the chain rule.

Polynomial Spline

1. Base functional specification (Linear, Quadratic, Cubic, Quartic, Polynomial)
2. Linear, Quadratic, Cubic, Quartic, Polynomial Basis Functions
    - Cubic Splines and Inflection Knots => Problem with the inflection knot points with cubic splines is that the "inflection" is now an extraneously supplied constraint, and in general may not be consistent with the C^2 criterion.
3. Segment interpolation relation
4. Control point analysis
5. Truth-ness specification
6. Smoothness constraint
7. Span-to-segment constraint transmission

8. Number of unknowns analysis
9. Solving for coefficients

Shape Preserving Tension Spline

1. 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)
2. 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.
3. 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).
4. 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.

5. 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.
6. 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.
7. 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
8. 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.

9. 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).
10. 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 $y_1 = \alpha_1$.
11. 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.
12. Rational Cubic Spline Coefficients:
    - $a = 1.y_0 + 0.y_1 + 0.y_0' + 0.y_0''$
    - $b = \lambda.y_0 + 0.y_1 + 1.y_0' + 0.y_0''$
    - $c = -\lambda.y_0 + 0.y_1 + \lambda.y_0' + \frac{1}{2}.y_0''$
    - $d = -1.y_0 + 1.y_1 + [-(1+\lambda)].y_0' + \left(-\frac{1}{2}\right).y_0''$
13. Rational Cubic Spline Derivatives:
    - $\alpha = a + bx + cx^2 + dx^3$
    - $\frac{d\alpha}{dx} = b + 2cx + 3dx^2$

- $\frac{d^2\alpha}{dx^2} = 2c + 6dx$
- $\beta = 1 + \lambda x(1-x)$
- $\frac{d\beta}{dx} = \lambda(1-2x)$
- $\frac{d^2\beta}{dx^2} = -\lambda$
- $\frac{dy}{dx} = \frac{\beta\frac{d\alpha}{dx} - \alpha\frac{d\beta}{dx}}{\beta^2}$
- $\frac{d^2y}{dx^2} = \frac{\beta^2\frac{d^2\alpha}{dx^2} - \alpha\beta\frac{d^2\beta}{dx^2} + 2\alpha\left(\frac{d\beta}{dx}\right)^2 - 2\beta\frac{d\alpha}{dx}\frac{d\beta}{dx}}{\beta^2}$

14. Designing λ_i for the Segment Infection/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.
    - Monotonizing Parameters for Rational Splines (Gregory (1984), Gregory (1986))

      => $\lambda_i = \mu_i + \left[f'(x_i) + f'(x_{i+1})\right]\frac{x_{i+1} - x_i}{y_{i+1} - y_i}$, again for $x_i < x < x_{i+1}$.
    - $\mu_i \geq -3$ makes it monotone in this segment.
    - $\mu_i = -2$ produces a rational quadratic.
    - Convergence is $\Theta\left(h^4\right)$ in all cases.
15. Co-convex choice for λ: A similar analysis can be done to make the spline co-convex, but the corresponding formulation requires a non-linear solution for λ_i.
16. Generalized Shape Controlling Interpolator: Given a pair of points $\{x_1, y_1\} \rightarrow \{x_2, y_2\} \Rightarrow \{0, y_1\} \rightarrow \{1, y_2\}$, 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.

17. Generically Partitioned Spline Derivative:

- $y(x) = \alpha(x)\beta(x)$
- $\frac{\partial y}{\partial x} = \frac{\partial \alpha}{\partial x}\beta + \alpha\frac{\partial \beta}{\partial x}$
- $\frac{\partial^2 y}{\partial x^2} = \frac{\partial^2 \alpha}{\partial x^2}\beta + 2\frac{\partial \alpha}{\partial x}\frac{\partial \beta}{\partial x} + \alpha\frac{\partial^2 \beta}{\partial x^2}$
- More generally $\frac{\partial^n y}{\partial x^n} = \sum_{r=0}^{n} {}^nC_r \frac{\partial^{n-r}\alpha}{\partial x^{n-r}}\frac{\partial^r \beta}{\partial x^r}$

18. Partitioned Interpolating Spline Coefficient: Given $\beta_0 = \beta_1 = 1$,

- $y_0 = \alpha_0$
- $y_1 = \alpha_1$
- $\left[\frac{\partial y}{\partial x}\right]_{x=0} = \left[\frac{\partial \alpha}{\partial x}\right]_{x=0}[\beta]_{x=0} + [\alpha]_{x=0}\left[\frac{\partial \beta}{\partial x}\right]_{x=0} \Rightarrow \left[\frac{\partial \alpha}{\partial x}\right]_0 = \left[\frac{\partial y}{\partial x}\right]_0 - \alpha_0\left[\frac{\partial \beta}{\partial x}\right]_0$
- 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:
  - $a = \{1\}.y_0 + \{0\}.y_1 + \{0\}.y'_0 + \{0\}.y''_0$
  - $b = \left[-\left|\frac{\partial \beta}{\partial x}\right|_0\right].y_0 + 0.y_1 + 1.y'_0 + 0.y''_0$
  - $c = \left[\left(\left|\frac{\partial \beta}{\partial x}\right|_0\right)^2 - \frac{1}{2}\left|\frac{\partial^2 \beta}{\partial x^2}\right|_0\right].y_0 + 0.y_1 + \left[-\left|\frac{\partial \beta}{\partial x}\right|_0\right].y'_0 + \frac{1}{2}.y''_0$
  - $c = \left[\frac{1}{2}\left|\frac{\partial^2 \beta}{\partial x^2}\right|_0 + \left|\frac{\partial \beta}{\partial x}\right|_0 - \left(\left|\frac{\partial \beta}{\partial x}\right|_0\right)^2 - 1\right].y_0 + 1.y_1 + \left[\left|\frac{\partial \beta}{\partial x}\right|_0 - 1\right].y'_0 + \left[-\frac{1}{2}\right].y''_0$

19. Interpolating Polynomial Splines of Degree n: 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 => $\frac{\partial^r y}{\partial x^r} = \sum_{i=0}^{n} \alpha_i \frac{i!}{(i-r)!} x^{i-r} = \sum_{i=r}^{n} \alpha_i \frac{i!}{(i-r)!} x^{i-r}$
- $\alpha_0 = y_0$
- $\alpha_r = \frac{1}{r!}\left[\frac{\partial^r y}{\partial x^r}\right]_{x=0}, \quad r \in [1, n-1]$
- $\alpha_n = y_1 - \sum_{i=0}^{n-1} \alpha_i = y_1 - \sum_{r=0}^{n-1} \left\{ \frac{1}{r!}\left[\frac{\partial^r y}{\partial x^r}\right]_{x=0} \right\}$

20. Polynomial Interpolating Spline Coefficient micro-Jack:

- $\frac{\partial \alpha_0}{\partial y_0} = 1, \frac{\partial \alpha_0}{\partial y_1} = 0, \left\{ \frac{\partial \alpha_0}{\partial\left(\left[\frac{\partial^r y}{\partial x^r}\right]\right)} \right\}_{x=0, 0 \neq r} = 0$
- $\frac{\partial \alpha_k}{\partial y_0} = 1, \frac{\partial \alpha_k}{\partial y_1} = 0, \left\{ \frac{\partial \alpha_k}{\partial\left(\left[\frac{\partial^r y}{\partial x^r}\right]\right)} \right\}_{x=0} = \frac{1}{r!}\delta_{kr}$
- $\frac{\partial \alpha_n}{\partial y_0} = -1, \frac{\partial \alpha_n}{\partial y_1} = 1, \left\{ \frac{\partial \alpha_n}{\partial\left(\left[\frac{\partial^r y}{\partial x^r}\right]\right)} \right\}_{x=0, 0 \neq r} = -\frac{1}{r!}$

21. 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.
- Interpolant and Tension Splines => Tension splines with $\sigma \neq 0$ can never be a polynomial order interpolant – only polynomial splines of order k (and degree $k-1$) are C^{k-2} continuous and interpolant!

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

23. Integrated Tension Spline Types: Both exponential and hyperbolic basis splines with a linear spline satisfy $\frac{\partial^2 y}{\partial x^2} - \sigma^2 y$.

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

- Hyperbolic Basis Splines: $\left\{1, x, \cosh\left(\frac{\sigma x}{x_{i+1} - x_i}\right), \sinh\left(\frac{\sigma x}{x_{i+1} - x_i}\right)\right\}$

24. Exponential Basis Functions:
    - Base Segment Formulation =>
        - $y = A + Bx + Ce^{\frac{\sigma x}{x_1 - x_0}} + De^{-\frac{\sigma x}{x_1 - x_0}}$
        - $y = \alpha + \beta\varepsilon + \gamma e^{\sigma\varepsilon} + \delta e^{-\sigma\varepsilon}$
    - Global <-> Local =>
        - $\alpha = A + Bx_0$
        - $\beta = B(x_1 - x_0)$
        - $\gamma = Ce^{\frac{\sigma x_0}{x_1 - x_0}}$
        - $\delta = De^{-\frac{\sigma x_0}{x_1 - x_0}}$
- Local <-> Global =>
    - $D = \delta e^{\frac{\sigma x_0}{x_1 - x_0}}$
    - $C = \gamma e^{-\frac{\sigma x_0}{x_1 - x_0}}$
    - $B = \frac{\beta}{x_1 - x_0}$
    - $A = \alpha - \frac{\beta x_0}{x_1 - x_0}$
- Co-efficient Calibration =>
    - $\alpha = y_0 - \frac{y_0''}{\sigma^2}$
    - $\gamma = \frac{1}{2}\left[\frac{y_0'' + \sigma(y_0' - \beta)}{\sigma^2}\right]$
    - $\delta = \frac{1}{2}\left[\frac{y_0'' - \sigma(y_0' - \beta)}{\sigma^2}\right]$

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

- Coefficient to Input Sensitivity Grid =>
  - $\alpha = [1]y_0 + [0]y_1 + [0]y_0' - \left[\dfrac{1}{\sigma^2}\right]y_0''$
  - $\beta = \left[\dfrac{-\sigma}{\sigma - \sinh\sigma}\right]y_0 + \left[\dfrac{\sigma}{\sigma - \sinh\sigma}\right]y_1 + \left[\dfrac{-\sinh\sigma}{\sigma - \sinh\sigma}\right]y_0' + \left[\dfrac{1 - \cosh\sigma}{\sigma(\sigma - \sinh\sigma)}\right]y_0''$
  - $\gamma = \left[\dfrac{1}{2(\sigma - \sinh\sigma)}\right]y_0 + \left[\dfrac{-1}{2(\sigma - \sinh\sigma)}\right]y_1 + \left[\dfrac{1}{2(\sigma - \sinh\sigma)}\right]y_0' + \left[\dfrac{\sigma - 1 + \cosh\sigma - \sinh\sigma}{2\sigma^2(\sigma - \sinh\sigma)}\right]y_0''$
  - $\delta = \left[\dfrac{-1}{2(\sigma - \sinh\sigma)}\right]y_0 + \left[\dfrac{1}{2(\sigma - \sinh\sigma)}\right]y_1 + \left[\dfrac{-1}{2(\sigma - \sinh\sigma)}\right]y_0' + \left[\dfrac{\sigma + 1 - \cosh\sigma - \sinh\sigma}{2\sigma^2(\sigma - \sinh\sigma)}\right]y_0''$

- Local Derivatives =>
  - $\dfrac{\partial y}{\partial \varepsilon} = \beta + \sigma\left[\gamma e^{\sigma\varepsilon} - \delta e^{-\sigma\varepsilon}\right]$
  - $\dfrac{\partial^2 y}{\partial \varepsilon^2} = \sigma^2\left[\gamma e^{\sigma\varepsilon} + \delta e^{-\sigma\varepsilon}\right]$
  - $\dfrac{\partial^3 y}{\partial \varepsilon^3} = \sigma^3\left[\gamma e^{\sigma\varepsilon} - \delta e^{-\sigma\varepsilon}\right]$
  - $\dfrac{\partial^r y}{\partial \varepsilon^r} = \beta\delta_{1r} + \sigma^r\left[\gamma e^{\sigma r} + (-1)^r \delta e^{-\sigma r}\right]$

- Global <-> Local Derivatives =>
  - $\dfrac{\partial y}{\partial x} = \dfrac{1}{(x_1 - x_0)}\dfrac{\partial y}{\partial \varepsilon}$
  - $\dfrac{\partial^2 y}{\partial x^2} = \dfrac{1}{(x_1 - x_0)^2}\dfrac{\partial^2 y}{\partial \varepsilon^2}$
  - $\dfrac{\partial^3 y}{\partial x^3} = \dfrac{1}{(x_1 - x_0)^3}\dfrac{\partial^3 y}{\partial \varepsilon^3}$

25. Hyperbolic Basis Functions:

- Base Segment Formulation =>

- $y = A + Bx + C\cosh\left(\frac{\sigma x}{x - x_0}\right) + D\sinh\left(\frac{\sigma x}{x - x_0}\right)$
- $y = \alpha + \beta\varepsilon + \gamma\cosh(\sigma\varepsilon) + \delta\sinh(\sigma\varepsilon)$

- Global <-> Local =>
  - $x = x_0 + \varepsilon(x_1 - x_0)$
  - $\varepsilon = \frac{x - x_0}{x_1 - x_0}$
  - $\alpha = A + Bx_0$
  - $\beta = B(x_1 - x_0)$
  - $\gamma = C\cosh\left(\frac{\sigma x_0}{x - x_0}\right) + D\sinh\left(\frac{\sigma x_0}{x - x_0}\right)$
  - $\delta = C\sinh\left(\frac{\sigma x_0}{x - x_0}\right) + D\cosh\left(\frac{\sigma x_0}{x - x_0}\right)$

- Coefficient to Input Sensitivity Grid =>
  - $\alpha = [1]y_0 + [0]y_1 + [0]y_0' + \left[\frac{-1}{\sigma^2}\right]y_0''$
  - $\beta = \left[\frac{\sigma}{\sigma - \sinh\sigma}\right]y_0 + \left[\frac{-\sigma}{\sigma - \sinh\sigma}\right]y_1 + \left[\frac{\sinh\sigma}{\sigma - \sinh\sigma}\right]y_0' + \left[\frac{\cosh\sigma - 1}{\sigma(\sigma - \sinh\sigma)}\right]y_0''$
  - $\gamma = [0]y_0 + [0]y_1 + [0]y_0' + \left[\frac{1}{\sigma^2}\right]y_0''$
  - $\delta = \left[\frac{-1}{\sigma - \sinh\sigma}\right]y_0 + \left[\frac{1}{\sigma - \sinh\sigma}\right]y_1 + \left[\frac{-1}{\sigma - \sinh\sigma}\right]y_0' + \left[\frac{1 - \cosh\sigma}{\sigma^2(\sigma - \sinh\sigma)}\right]y_0''$

- Local Derivatives =>
  - $\frac{\partial y}{\partial \varepsilon} = \beta + \sigma[\gamma\sinh(\sigma\varepsilon) + \delta\cosh(\sigma\varepsilon)]$
  - $\frac{\partial^2 y}{\partial \varepsilon^2} = \sigma^2[\gamma\cosh(\sigma\varepsilon) + \delta\sinh(\sigma\varepsilon)]$
  - $\frac{\partial^3 y}{\partial \varepsilon^3} = \sigma^3[\gamma\sinh(\sigma\varepsilon) + \delta\cosh(\sigma\varepsilon)]$

- $\frac{\partial^r y}{\partial \varepsilon^r} = \sigma^r \left[\gamma \cosh(\sigma\varepsilon) + \delta \sinh(\sigma\varepsilon)\right]$ if r is even.
- $\frac{\partial^r y}{\partial \varepsilon^r} = \beta\delta_{1r} + \sigma^r \left[\gamma \sinh(\sigma\varepsilon) + \delta \cosh(\sigma\varepsilon)\right]$ if r is odd.

- Global <-> Local Derivatives =>
  - $\frac{\partial y}{\partial x} = \frac{1}{(x_1 - x_0)} \frac{\partial y}{\partial \varepsilon}$
  - $\frac{\partial^2 y}{\partial x^2} = \frac{1}{(x_1 - x_0)^2} \frac{\partial^2 y}{\partial \varepsilon^2}$
  - $\frac{\partial^3 y}{\partial x^3} = \frac{1}{(x_1 - x_0)^3} \frac{\partial^3 y}{\partial \varepsilon^3}$

26. Segment interpolation relation – reduction onto linear/cubic spline
27. Switch between linear and cubic spline
28. Alternate specifications of the segment interpolation (Trojand (2011))
29. Localized and normalized tension (Trojand (2011))
    - 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)).
30. <u>Problems with Hyperbolic/Tension Splines</u>:
    - 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)).
    - They have been gradually pushed out by ν splines (Nielson (1974)) and rational splines.

Optimizing Spline Basis Function Jacobian

1. Coefficient- Value Micro-Jacobian: $FA = Y$ where
   - A is the matrix of the basis coefficients $\{a_0, a_1, a_2, ..., a_{r+1}, ..., a_{k+1}, ..., a_{n-1}\}$
   - Y is the matrix (column valued) of the values (RHS). In particular, it is the boundary segment calibration nodal values in the following order:

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

   - F is the matrix of the coefficients of the basis function values and their derivatives. It is the following 2D Matrix:
     - $l = 0; j = 0, ...n-1 \Rightarrow F_{0j} = f_j(x=0)$
     - $l = 1; j = 0, ...n-1 \Rightarrow F_{1j} = f_j(x=1)$
     - $2 \le l \le k+1; j = 0, ...n-1 \Rightarrow F_{lj} = \left[\frac{\partial^{l-1} f_j}{\partial x^{l-1}} \right]_{x=0}$
     - $l \to k+2, ..., n-1; j \to 0, ..., n-1 \Rightarrow Q_{l,j}$ where

       $$Q_{l,j} = \int_{x_i}^{x_{i+1}} \left[\frac{\partial^m f_l(x)}{\partial x^m} \right] \left[\frac{\partial^m f_j(x)}{\partial x^m} \right] dx.$$

2. Coefficient-Value Micro-Jacobian: Given $FA = Y$, the coefficient-value micro-Jack is $\frac{\partial a_i}{\partial y_j} = \left[F^{-1} \right]_{ij}$.

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:

$$\Lambda\left(\hat{\mu}, k, m, n, p, \vec{\lambda}\right) = \sum_{i=1}^{n-1}\left\{\left[Y_i - \hat{\mu}_P(x_i)\right]^2 + \lambda_i \int_{x_1}^{x_{i+1}}\left[\frac{\partial^m \hat{\mu}_P(x)}{\partial x^m}\right]^2 dx\right\} + \left[Y_n - \hat{\mu}_P(x_n)\right]^2$$

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 $\Lambda\left(\hat{\mu}, k, m, n, p, \vec{\lambda}\right)$ formulated earlier, we get $\Lambda_{NM}\left(\hat{\mu}, k, m, n, p, \vec{\lambda}\right) = \sum_{i=1}^{n-1}\left\{\lambda_i \int_{x_1}^{x_{i+1}}\left[\frac{\partial^m \hat{\mu}_P(x)}{\partial x^m}\right]^2 dx\right\}$ where $\Lambda_{NM}\left(\hat{\mu}, k, m, n, p, \vec{\lambda}\right)$ 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 $\Lambda_i = \int_{x_i}^{x_{i+1}}\left[\frac{\partial^m \hat{\mu}(x)}{\partial x^m}\right]^2 dx$.

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

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

6. Generic Coefficient Constrained Optimization Setup: $\frac{\partial \Lambda_i}{\partial \alpha_{ik}} = 0 \Rightarrow \sum_{j=0}^{n-1} \alpha_{ik} Q_{ijk} = 0$

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

7. Polynomial Formulation for $\Lambda_{NM}\left(\hat{\mu}, k, m, n, p, \vec{\lambda}\right)$: 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 $\Lambda_{NM}\left(\hat{\mu}, k, m, n, p, \vec{\lambda}\right)$ on a per-segment basis by re-casting $\Lambda_{NM}\left(\hat{\mu}, k, m, n, p, \vec{\lambda}\right)$ to $\Lambda_i(k, m, p) = \int_{x_1}^{x_{i+1}} \left[\frac{\partial^m \hat{\mu}_i(p, x)}{\partial x^m}\right]^2 dx$.

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

$$\Lambda_i(k, m, p) = \int_{x_1}^{x_{i+1}} \left[\sum_{j=m}^{p} \frac{j!}{(j-m)!} \alpha_{ij} x^{j-m}\right]^2 dx = \sum_{j=m}^{p}\sum_{l=m}^{p} \left\{\frac{j!}{(j-m)!}\frac{l!}{(l-m)!} \alpha_{ij}\alpha_{il}\left[\frac{x_{i+1}^{\,j+l-2m+1} - x_i^{\,j+l-2m+1}}{j+l-2m+1}\right]\right\}$$

.

10. Minimization of $\Lambda_i(k, m, p)$: $\frac{\partial \Lambda_i(k, m, p)}{\partial \alpha_{ij}} = 0 \Rightarrow 2\overline{\alpha_{iq}}\beta_{qq} + \sum_{j=m, j\neq q}^{p} \alpha_{ij}\beta_{qj} = 0$,

$m \le j \le p$.

   - Here $\beta_{qj} = \frac{q!}{(q-m)!}\frac{j!}{(j-m)!}\left\{\frac{x_{i+1}^{\,q+j-2m+1} - x_i^{\,q+j-2m+1}}{q+j-2m+1}\right\}$.

   - $\overline{\alpha_{iq}} = -\frac{1}{\beta_{qq}} \sum_{l=m, l\neq q}^{p} \alpha_{il}\beta_{ql}$.

- Since $\frac{\partial^2 \Lambda_i(k,m,p)}{\partial \alpha_{ij}^{\ 2}} = \beta_{qq}$ and $\beta_{qq} > 0$, $\overline{\alpha_{iq}}$ corresponds to the minimum of $\Lambda_i(k,m,p)$.
- Thus, if $x_i = 0$ and $x_{i+1} = 1$, β_{qj} becomes

$$\beta_{qj} = \frac{q!}{(q-m)!}\frac{j!}{(j-m)!}\left\{\frac{1}{q+j-2m+1}\right\}.$$

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 $k+1 \le q < p$ 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 $k_{input} < p-2$, 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.

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 Curvature Penalty Estimator (DCPE): Using Figure 7), we now define DCPE as $DCPE = \frac{Area_{ShadedPart}}{Area_{ABCD}} = \frac{\int_{x_i}^{x_{i+1}} \left[\frac{\partial^m \hat{\mu}(x)}{\partial x^m}\right]^2 dx}{\max\left\{\left[\frac{\partial^m \hat{\mu}(x)}{\partial x^m}\right]^2\right\}(x_{i+1} - x_i)}$
4. Pros/Cons of the above Representation of DCPE: If the basis functions have near-delta functional forms (Figure 8), DCPE will still remain ≈ 0, and the metric is not very meaningful in that case. Fortunately, such delta-type basis functions are rare.
5. Aggregate DCPE Measure: Need a consolidated DCPE metric that spans across all the segments in a span, i.e., the span DCPE.

Bernstein Polynomial

1. Bernstein Polynomial of degree n, and term ν: $b_{\nu,n}(x) = {}^nC_\nu x^\nu (1-x)^{n-\nu}$ where $\nu = 0,...,n$.
   - Bernstein Polynomial Convenience Re-cast #1: $b_{\nu,n}(x) = n! \frac{x^\nu}{\nu!} \frac{(1-x)^{n-\nu}}{(n-\nu)!}$.
   - Bernstein Polynomial Convenience Re-cast #2: $P_{b,c}(x) = (b+c)! F(x,b) F(1-x,c)$ where $F(x,b) = \frac{x^b}{b!}$.

2. Derivative of the Bernstein Polynomial: $\frac{d^r b_{\nu,n}(x)}{dx^r} = n\left[\frac{d^{r-1} b_{\nu-1,n}(x)}{dx^{r-1}} - \frac{d^{r-1} b_{\nu-1,n-1}(x)}{dx^{r-1}}\right]$.

   - Bernstein Polynomial Re-cast #2 Derivative:

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

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 $x = 1$.

   - Further, $b_{0,i} = 1$ for $0 \le x < 1$, and $b_{0,i} = 0$ otherwise.

Other Tension Splines

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, $f_i(x) = p_i\left[q_i^{-1}(x)\right]$ 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: $f_i'(x_i) = d_i$; further, we 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 $\lambda_i = \mu_i = 1$, $q_i(x) = x$, thus f_i becomes cubic. Also if $\lambda_i = \mu_i = 0$, f_i reduces to linear.

Smoothing Splines

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").
   - In the case of non-Gaussian priors, the smoothing estimation process is called the Generalized Linear Model.
3. Smoothing Spline Formulation: Given $x_1 < x_2 < ... < x_n$, and the function μ that fits the points $[x_i, Y_i]$ from $Y_i = \mu(x_i)$. 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. 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
7. 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

8. 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.
9. Extension to k-Curvature Penalty: For the case where k > 2, we would need to choose a k+1 degree spline to retain linearity of the segment k-curvature – therefore, a k+1 degree spline is the k-curvature penalty minimizer spline. This also preserves the C^k continuity constraint.
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: $\int_{x_1}^{x_n}\left[\frac{\partial^k \hat{\mu}(x)}{\partial x^k}\right]dx = c$.

Density Smoothing

1. 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.

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.
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. Polynomial Regression Splines: This does 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.

Multi-dimensional Splines

1. Symmetrical Multi-dimensional variates: The trivial univariate ordering $x_1 < x_2 < ... < x_n$ needs revising in the context of certain multivariates, e.g., symmetrical multivariates.
   - A general "distance from focal node" t_i makes to more sense to set in the ascending order. Thus $t_i = \sqrt{(x_i - x_F)^2 + ... + (z_i - z_F)^2}$, where $[x_F, ..., z_F]$ are the multivariate nodes corresponding to the focal node.
   - Use Cartesian/radial/axial basis functions to formulate the segments in terms of the surface vector coefficients in "symmetrical variate" situations.
2. Surface Energy Minimization: Surface energy minimization using the "sigma" tension parameter – formulate equation.
   - 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.
3. 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

$$\hat{\mu}(x_1, x_2) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \beta_{ij} B_i(x_1) B_j(x_2).$$

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**.
- Delbourgo, R. (1989): Shape preserving interpolation to convex data to rational functions with quadratic numberator and linear denominator, *IMA J. Numer. Anal.* **9**: 123-136.
- Delbourgo, R., and J. Gregory (1983): C^2 rational spline quadratic interpolation to monotonic data, *IMA J. Numer. Anal.* **3**: 141-152.
- Delbourgo, R., and J. Gregory (1985a): The determination of the derivative parameters for a monotonic rational quadratic interpolant, *IMA J. Numer. Anal.* **5**: 397-406.
- Delbourgo, R., and J. Gregory (1985b): Shape preserving piecewise rational interpolation, *SIAM J. Sci. Stat. Comput.* **6**: 967-976.
- 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.
- Foley, T. (1988): A Shape preserving Interpolant with Tension Controls, *Computer Aided Geometric Design* **5**.
- Goodman, T. (2002): *Shape preserving interpolation by curves*.
- Gregory, J. (1984): Shape preserving rational spline interpolation, in *Rational Approximation and Interpolation*, Graves-Morris, Saff, and Varga (Eds.), **Springer-Verlag**, 431-441.
- Gregory, J. (1986): Shape preserving spline interpolation, *Computer Aided Design* **18**: 53-58.
- Judd, K. (1998): *Numerical Methods in Economics*. **MIT Press**.

- Kaklis, P. D., and D. G. Pandelis (1990): Convexity preserving polynomial splines of non-uniform degree, *IMA J. Numer. Anal.* **10**: 223-234.
- Katz, M. (2011): *Multi-variable Analysis: A Practical Guide for Clinicians and Public Health Researchers*. **Cambridge University Press**.
- Lamberti, P., and C. Manni (2001): Shape preserving C^2 functional interpolation via parametric cubics, *IMA J. Numer. Anal.* **28**: 229-254.
- 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 comonotone 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 (Eds.), **World-Scientific**, 211-228.
- Manni, C. (2001): On shape preserving C^2 Hermite interpolation, *BIT.* **14**: 127-148.
- Manni, C., and P. Sablonniere (1997): Monotone interpolation of order 3 by C^2 cubic splines, *IMA J. Numer. Anal.* **17**: 305-320.
- Manni, C., and M. L. Sampoli (1998): Comonotone parametric Hermite interpolation, in *Mathematical Methods for Curves and Surfaces II,* Daehlen, Lyche, and Schumaker (Eds.), **Vanderbilt University Press**, 343-350.
- McAllister, D., E Passow, and J Roulier (1977): Algorithms for computing shape preserving spline interpolation to data. *Math. Comp.* **31**: 717-725.
- Nielson, G. (1974): Some Piecewise Polynomial Alternatives to Splines under Tension, in *Computer Aided Geometric Design,* R. Barnhill, R. Reisenfeld (Eds.), **Academic Press**, 209-235.
- Passow, E., and J Roulier (1977): Monotone and convex interpolation. **Society for Industrial and Applied Mathematics**, *J. Numer. Anal.* **14**: 904-909.
- Preuss, 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.
- Runge's phenomenon (Wiki): *Wikipedia Entry for Runge's phenomenon*.
- Sapidis, N., P Kaklis, and T Loukakis (1988): A Method for computing the Tension Parameters in Convexity preserserving Spline-in-Tension Interpolation, *Numer. Math* **54**: 179-192.
- Schoenberg, I. (1946): Contributions to the problem of approximation of equi-distant data by analytic functions, *Quart. Appl. Math.* **4**: 45-99, and 112-141.
- Schweikert, D. (1966): An interpolation curve using a spline in tension, *J. Math. Phys.* **45**: 312-317.
- Spath, H. (1969): Exponential Spline Interpolation, *Computing* **4**: 225-233.
- Spath, H. (1974): *Spline Algorithms for Curves and Surfaces*. **Utilitas Mathematica Pub. Inc**. Winnipeg.
- Spline (Wiki): *Wikipedia Entry for Spline*.
- Trojand, D. (2011). *Splines Under Tension*.

Spline Library Software Components

Functionality behind Spline Library is available across 3 packages – univariate function package, the Span/Segment package, and the Spline Basis function set package.

Univariate Function Package (org.drip.math.function)

The univariate function package implements the individual univariate functions, their convolutions, and reflections. It contains the following classes/interfaces:

- AbstractUnivariate: This abstract class provides the evaluation of the given basis/objective function and its derivatives for a specified variate. Default implementations of the derivatives are for black-box, non-analytical functions.
- BernsteinPolynomial: This class provides the evaluation of Bernstein polynomial and its derivatives for a specified variate. The degree exponent specifies the order of the Bernstein polynomial.
- ExponentialTension: This class provides the evaluation of exponential tension basis function and its derivatives for a specified variate. It can be customized by the choice of exponent, the base, and the tension parameter.
- HyperbolicTension: This class provides the evaluation of hyperbolic tension basis function and its derivatives for a specified variate. It can be customized by the choice of the hyperbolic function and the tension parameter.
- NaturalLogSeriesElement: This class provides the evaluation of a single term in the expansion series for the natural log. The exponent parameter specifies which term in the series is being considered.
- Polynomial: This class provides the evaluation of the n^{th} order polynomial and its derivatives for a specified variate. The degree n specifies the order of the polynomial.

- RationalShapeControl: This class provides the evaluation of the rational shape control spline basis described above. The tension parameter set in the constructor customizes the spline.
- UnivariateConvolution: This class provides the evaluation of the point value and the derivatives of the convolution of 2 univariate functions for a specified variate.
- UnivariateReflection: For a given variate x, this class provides the evaluation and derivatives of the reflection at $1 - x$.

Segment/Span Layout Package (org.drip.math.grid)

This package implements the layout of the n-D grid functionality in accordance with the calibration schema set out earlier.

- Inelastics: This class the inelastic fields of the given segment – in this case the start/end co-ordinates.
- Segment: This abstract class extends Inelastics, and incorporates segment specific inelastic parameters. Interpolating segment spline functions and their coefficients are implemented/calibrated in the overriding spline classes. It provides functionality for assessing the various segment attributes:
    - Segment Monotonicity.
    - Interpolated Function Value, the ordered derivative, and the corresponding Jacobian.
    - Segment Local/Global Derivative.
    - Evaluation of the Segment Micro-Jack.
    - Head / Regular Segment calibration - both of the basis function coefficients and the Jacobian.
- SegmentControlParameters: This class holds the parameters the guide the creation and the behavior of the segment. It holds the segment elastic/inelastic parameters and the named basis function set.

- SegmentMonotonicity: This class contains the monotonicity details related to the given segment. Indicates whether the segment is monotonic, and if not, whether it contains a maximum, a minimum, or an inflection.
- Span: This class implements the span that spans multiple segments. It holds the ordered segment sequence, the segment control parameters, and, if available, the spanning Jacobian. It exports the following group of functionality:
    - Construct adjoining segment sequences in accordance with the segment control parameters
    - Calibrate according to a varied set of (i.e., NATURAL/FINANCIAL) boundary conditions
    - Interpolate both the value, the ordered derivatives, and the Jacobian at the given ordinate
    - Compute the monotonicity details - segment/span level monotonicity, co-monotonicity, local monotonicity.
    - Insert knots

    It also exports several static span creation/calibration methods to generate customized basis splines, with customized segment behavior using the segment control.

Basis Spline Package (org.drip.math.spline)

This package implements the basis set across the different splines – their creation, the segment level calibration, the customization, and segment-level inference values.

- BasisSetParams: This stub class holds out per-segment basis set parameters.
- ExponentialTensionBasisSetParams: This class implements per-segment parameters for the exponential tension basis set - currently it only contains the tension parameter.
- KaklisPandelisTensionBasisSetParams: This class implements per-segment parameters for the Kaklis-Pandelis basis set - currently it only holds the polynomial tension degree.

- PolynomialBasisSetParams: This class implements per-segment basis set parameters for the polynomial basis spline - currently it holds the number of basis functions.
- SegmentBasisSetBuilder: This class implements the basis set and spline builder for the following types of splines:
  - Exponential basis tension splines
  - Hyperbolic basis tension splines
  - Polynomial basis splines
  - Bernstein Polynomial basis splines
  - Kaklis Pandelis basis tension splines

  The elastic coefficients for the segment using C^k basis splines inside $[0,...,1)$ - globally $[x_0,...,x_1)$ are extracted for $y = Interpolator\left(C^k, x\right) * ShapeController\left(x\right)$ where x is the normalized ordinate mapped as: $x \to \frac{x - x_{i-1}}{x_i - x_{i-1}}$. The inverse quadratic/rational spline is a typical shape controller spline used.
- SegmentCk: This concrete class extends segment, and implements the segment's C^k based spline functionality. It exports the following:
  - Calibration => Head Calibration, Regular Calibration
  - Estimated Segment Elastics => The Basis Functions and their coefficients, C^k, the shape controller
  - Local Point Evaluation => Value, Ordered Derivative
  - Local Monotonicity
  - Local coefficient/derivative micro-Jack, and value/coefficient micro-Jack
  - Local Jacobians => Value Micro Jacobian, Value Elastic Jacobian, Composite Value Jacobian
- SegmentConstraint: This class holds the segment coefficient constraints and their values.
- SegmentInelasticParams: This class implements basis per-segment elastics parameter set. Currently it contains C^k and the segment specific constraints.

Figure #1
SEGMENT/SPAN Structure Layout

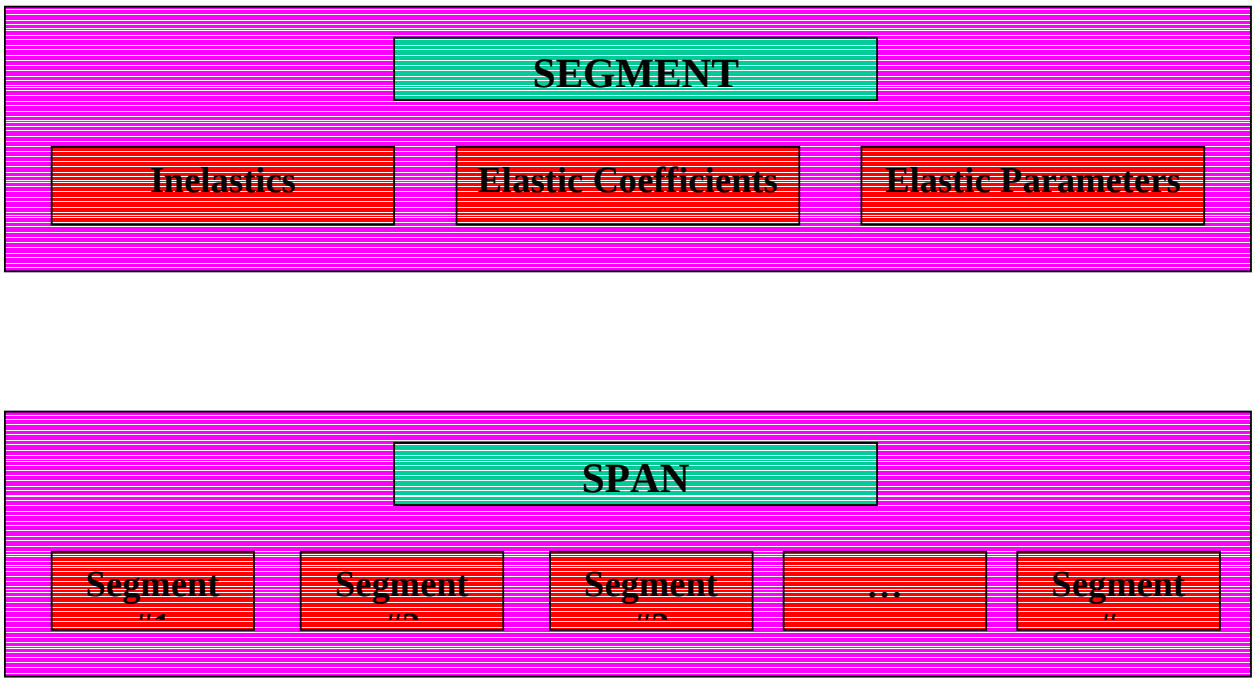

Figure #2
BASIS SPLINE HIERARCHY

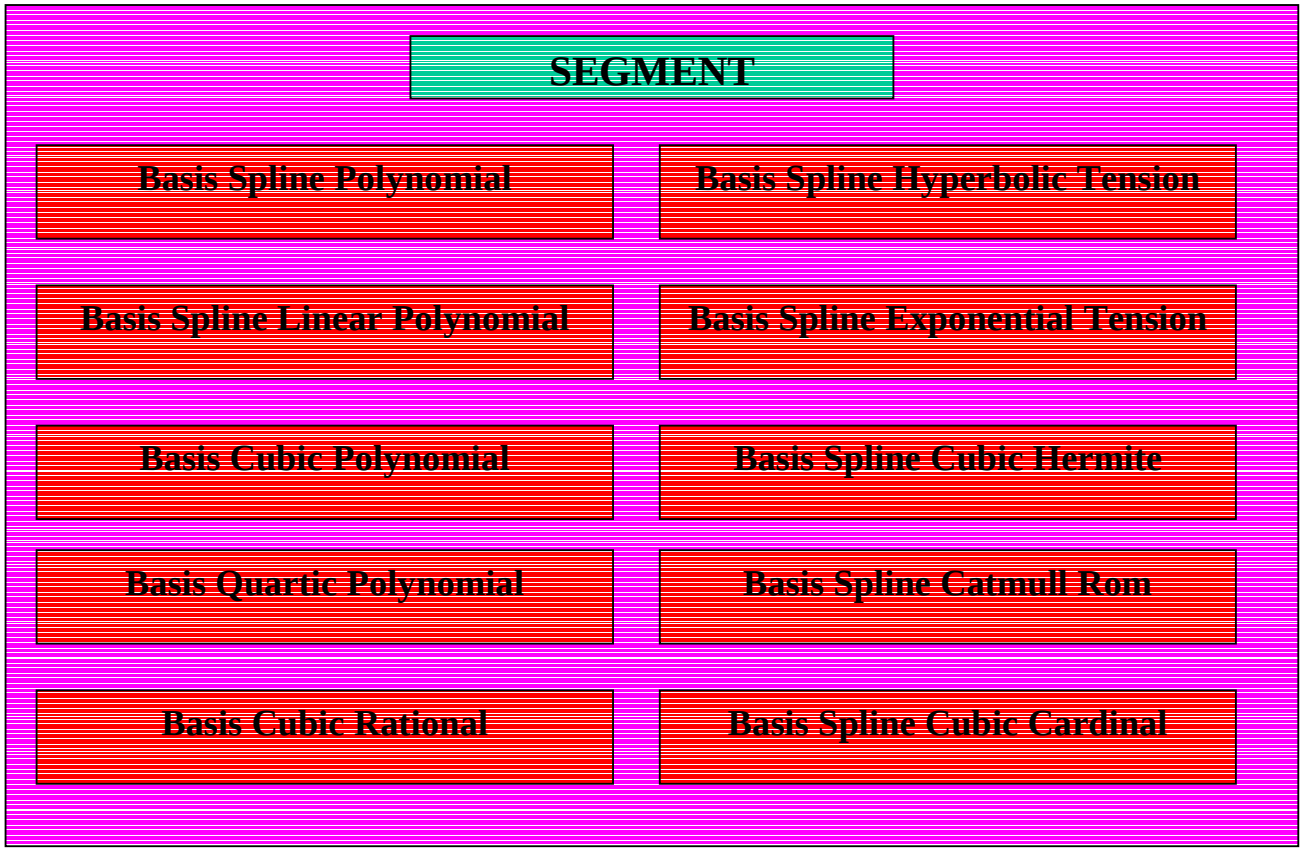

Figure #3
THIRD ORDER SECOND DEGREE SPLINE

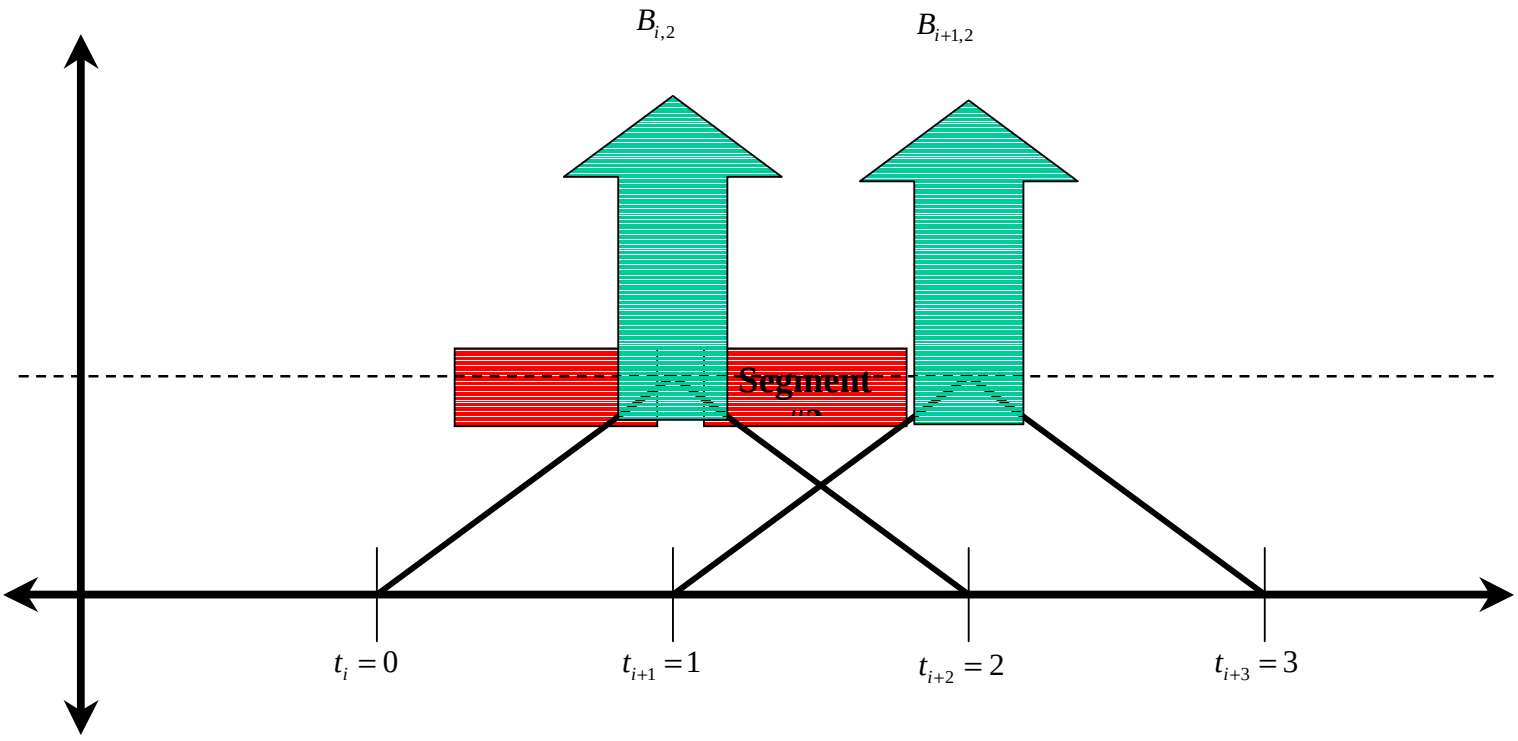

Figure #4
B SPLINE INTERPLATION SCHEME

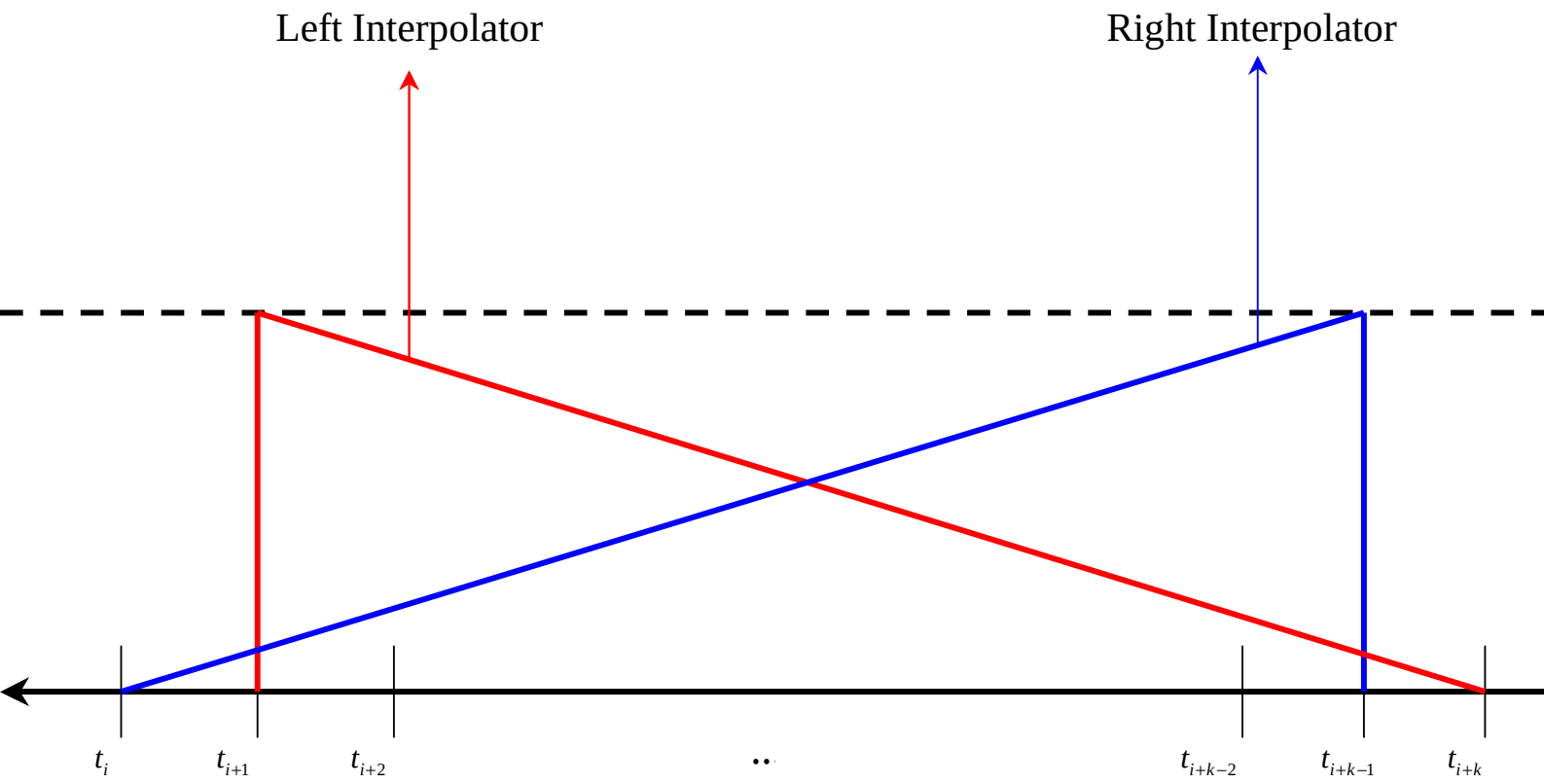

Figure #5
PENALTY MINIMIZER METRIC - #1

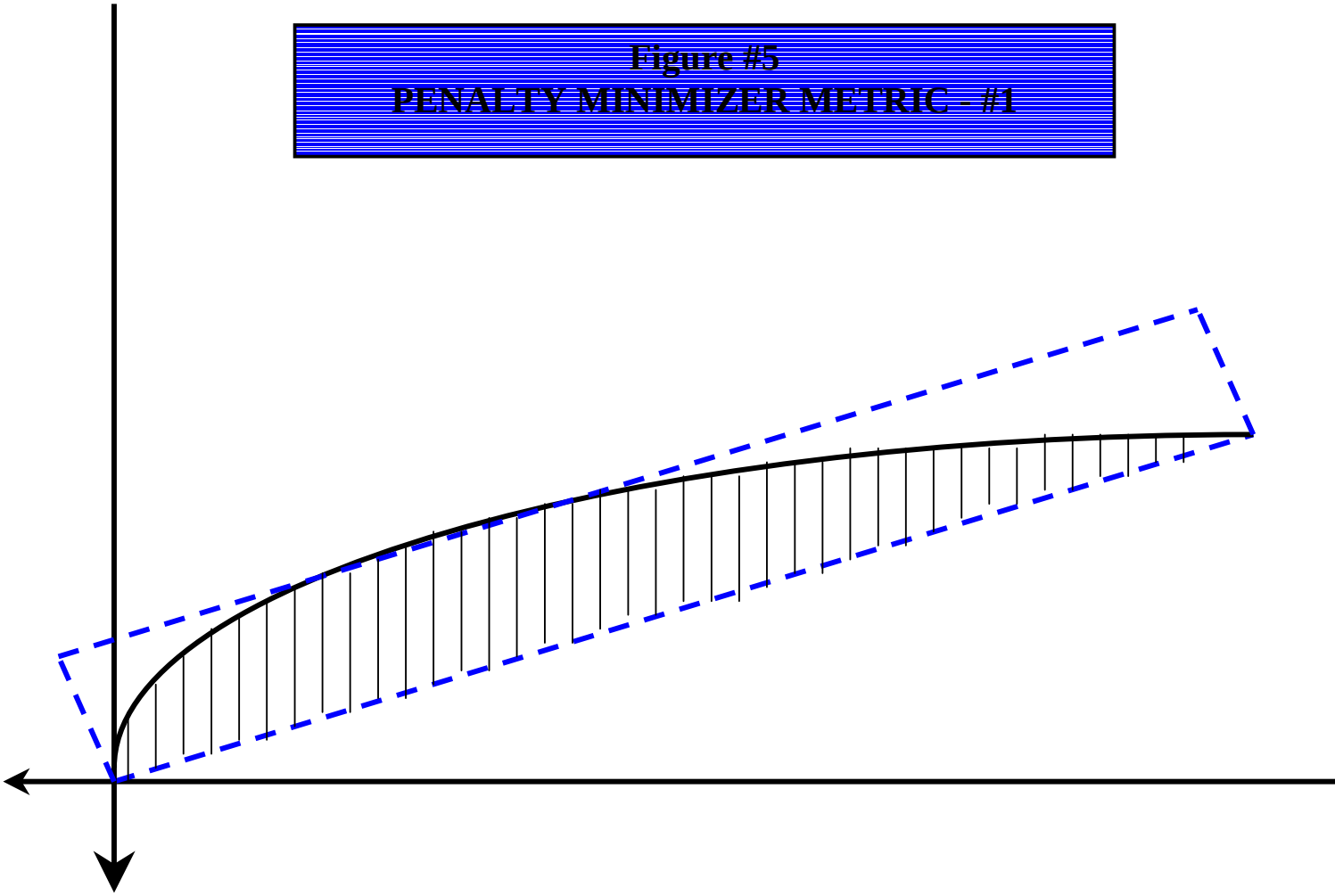

Figure #6
PENALTY MINIMIZER METRIC - #2

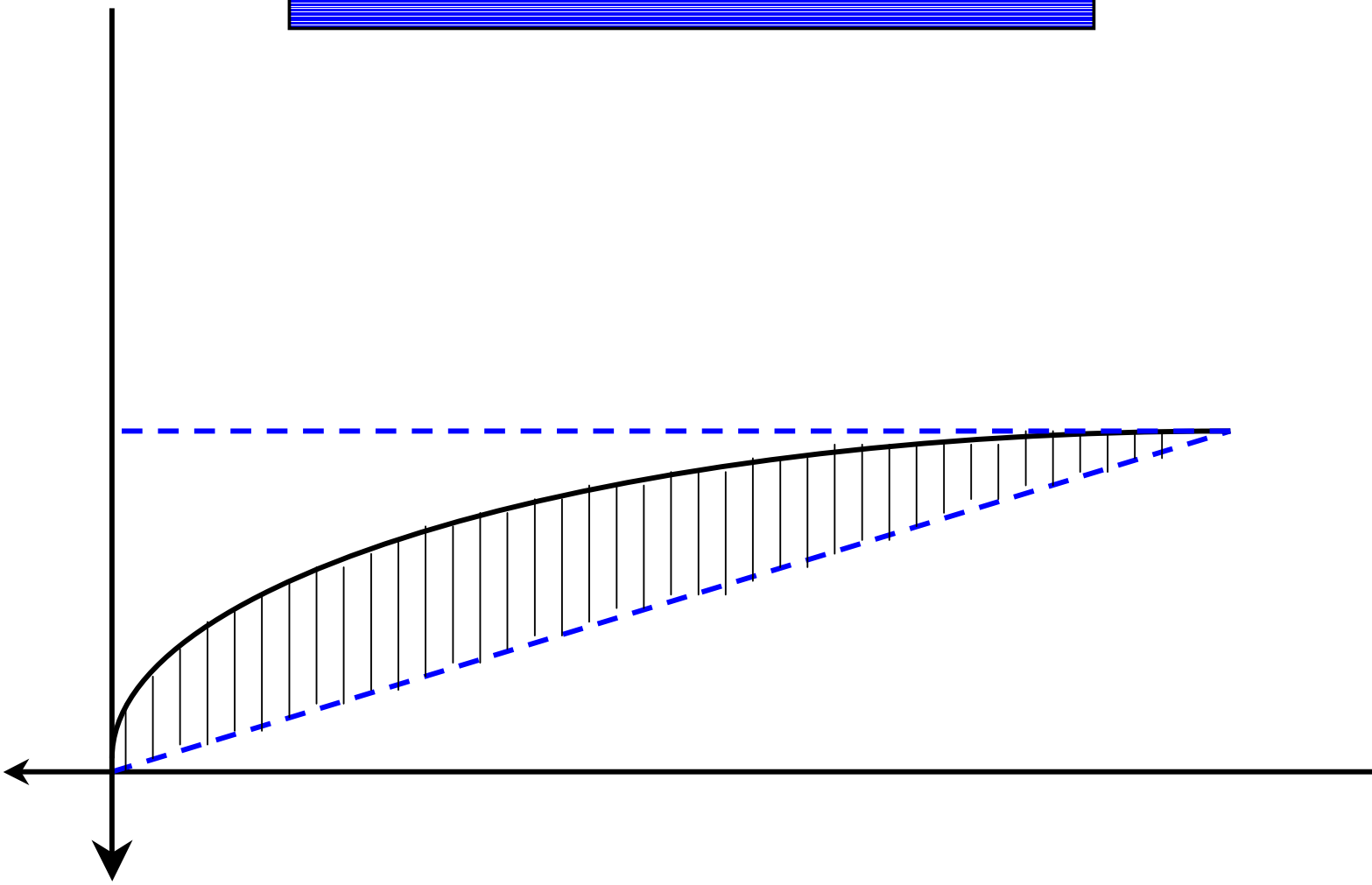

Figure #7
DIMENSION-LESS CURVATURE PENALTY ESTIMATOR

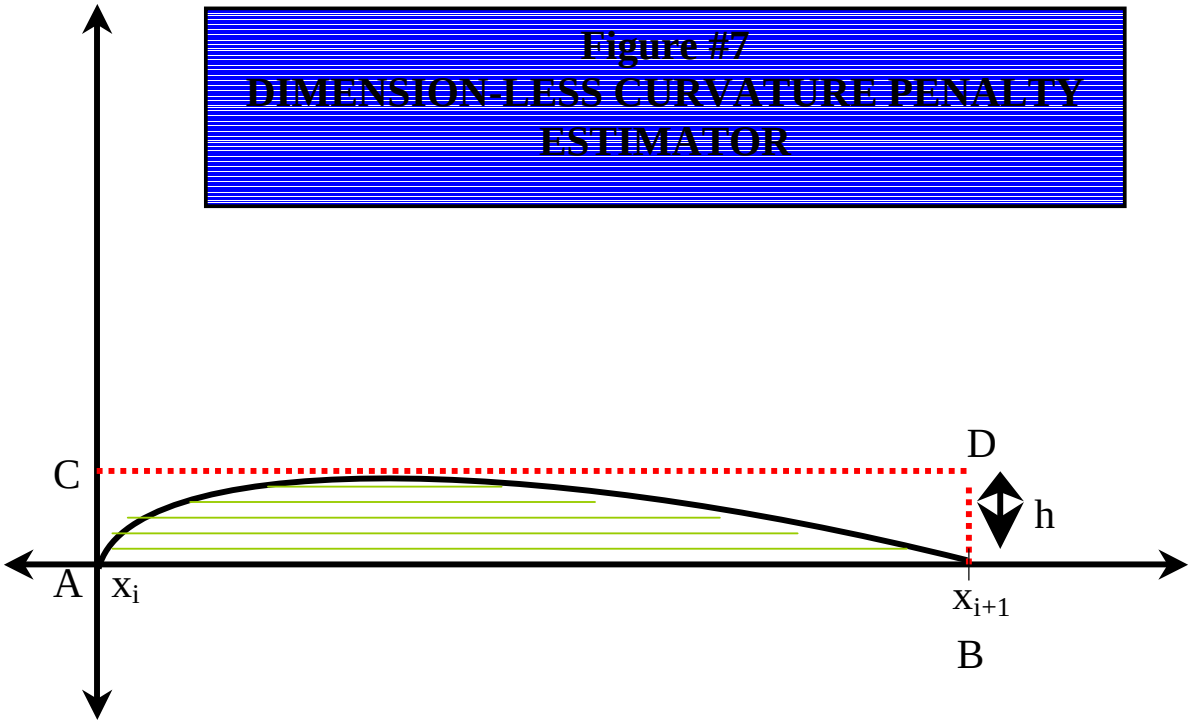

Figure #8
NEAR-DELTA DCPE