

A guide to using the Addix.Spline curve approximation and interpolation library

Stephen Weston

Addix Software Consultancy Limited

Version 1.6

September 2002

Contents

1. Installation, system requirements, access and pricing
2. AutoTensionSpline
3. BasisSpline
4. BezierCurve
5. BezierCurveDerivatives
6. BernsteinBasis
7. BezierBasisPoint
8. BezierCoefficient
9. CatmullRomSpline
10. GSpline
11. MultipleOpenKnotVector
12. NonUniformOpenKnotVector
13. OpenKnotBasisSplineVector
14. OpenKnotVector
15. OpenUniformKnotBasisSplineCurve
16. OpenUniformKnotRBasisSplineCurve
17. QuinticSpline
18. RationalBasisSplineOpenKnotVector

19. TensionSpline
20. UniformOpenBasisSplineCurve
21. UniformPeriodicKnotVector
22. XSpline

1 Installation and system requirements

1.1 Installation

The library is installed by downloading and executing the setup program available from the Addix website at www.addix.com. The library will work with versions of Excel and Visual Basic for Applications later than Excel-97, as well as with versions of Visual Basic later than version 5. Once installed, the example spreadsheet contains invocations of all functions in both their Excel form and their VBA form.

1.2 System requirements

There are no specific system requirements for using the Spline library. It is, however, likely that superior performance will be obtained if the machine running the Library is at least a Pentium 3, 400Mhz machine with at least 64Mb of RAM. The machine on which the library was built and tested was a Pentium 3, 800Mhz machine with 256Mb of RAM. This configuration produced extremely fast execution over even the largest of datasets. It is worth bearing in mind that larger resources of video RAM will also improve performance.

1.3 Access and pricing

It is the policy of Addix to provide permanent free access to a single function in all of its libraries as a means of allowing potential users to evaluate the product at the leisure. The Spline Library is no exception to this and the free function is the BasisSpline function which calculates an interpolation spline using a third order polynomial as the basis function. Access to the remaining functions in the library is only granted upon receipt of cleared funds to cover the appropriate charge for the relevant numbers of licences purchased. Contact Addix as follows:

Visit our website at: www.addix.com

Email us at: sales@addix.com

Call our office on: 07020-939710

2 AutoTensionSpline

AutoTensionSpline allows interpolation using splines under automatically generated non-uniform tension.

Usage

AutoTensionSpline(*Independent, Dependent, Required,*
[Curve, SlopeStart, SlopeEnd, Tension])

Parameters

Independent - a contiguous single column array of monotonically increasing values. Must have the same number of observations as the *Dependent* array.

Dependent - a contiguous single column array of values. Must have the same number of values as the *Independent* array.

Required - contiguous single column array containing one or more independent values for which the splined dependent value is required. The values must be ordered in same direction as the independent values.

CalculateOption - a single integer value as follows:

Setting *CalcOption* value to 1 does the following:

- a) Sets the number of continuous derivatives to 1 at each knot point;
- b) Fits first and last three points using a monotonicity constrained parabolic fit - equivalent to a C^1 fit.

- c) Fits non-periodic spline;

- d) Forces tensions to be calculated for each pair of points.

Setting *CalcOption* value to 2 does the following:

- a) Sets the number of continuous derivatives to 2 at each knot point;
- b) Fits first and last three points using a monotonicity constrained parabolic fit - equivalent to a C^1 fit;

- c) Fits non-periodic spline;

- d) Forces tensions to be calculated for each pair of points.

Setting *CalcOption* value to 3 does the following:

- a) Sets the number of continuous derivatives to 1 at each knot point;
- b) Fits slope of first and last knot points iteratively to maintain local convexity and monotonicity

- c) Fits non-periodic spline;

- d) Forces tensions to be calculated for each pair of points.

Setting *CalcOption* value to 3 does the following:

- a) Sets the number of continuous derivatives to 2 at each knot point;
- b) Fits slope of first and last knot points iteratively to maintain local convexity and monotonicity

- c) Fits non-periodic spline;

- d) Forces tensions to be calculated for each pair of points.

ReturnOption - a single integer value as follows:

Setting *ReturnOption* to a value of 1 does the following:

Returns the derivatives at the knot points. If used in conjunction with *CalcOption* set to 1, this will return the vector of first derivatives at the knot points. If used in conjunction with *CalcOption* set to 2, this will return the vector of second derivatives at the knot points.

Setting *ReturnOption* to a value of 2 does the following:

Returns the values of the tension parameters at the knot points. If used in conjunction with *CalcOption* set to 1, this will return the vector of tension

factors at the knot points associated with the first derivatives. If used in conjunction with `CalcOption` set to 2, this will return the vector tension factors associated with the second derivatives at the knot points.

Setting `ReturnOption` to a value of 3 does the following:

Returns the number of iterations required to achieve the requested fit.

Setting `ReturnOption` to a value of 4 does the following:

Returns the values of the splined function at the requested dependent value points.

Setting `ReturnOption` to a value of 5 does the following:

Returns the values of the splined function at the requested dependent value points merged with the original dependent and independent data arrays.

Remarks

This spline technique is based on the approach of Renka to producing shape-preserving interpolating splines using piece-wise exponential functions which allow potentially different tension factors to be associated with each interval. The construction allows the efficient computation of the tension factor for each interval. It is possible to have either C^1 or C^2 interpolants. An explanation of the mathematical theory of auto-tension splines can be found in the reference document which forms part of this product.

3 BasisSpline

`BasisSpline` allows interpolation using a spline constructed using a third order polynomial basis function.

Usage

BasisSpline(*Independent, Dependent*)

Parameters

Independent - a contiguous single column array of monotonically increasing values. Must have the same number of observations as the *Dependent* array.

Dependent - a contiguous single column array of values. Must have the same number of values as the *Independent* array.

Remarks

Basis Splines can be constructed using a polynomial of virtually order. The implementation of basis splines in this library is limited to second order (quadratic) and third order (cubic) basis functions. Basis splines do not interpolate any knot points and so are only capable of being approximating splines.

4 BernsteinBasis

`BernsteinBasis` returns the basis coefficients for a given array and polynomial of specified degree.

Usage

BernsteinBasis(*Independent, Degree*)

Parameters

Independent - a contiguous single column array of monotonically increasing values.

Degree - the required degree of the basis polynomial.

Remarks

The function returns an array of the BernsteinBasis polynomial coefficients:

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad (1)$$

with

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \quad (2)$$

5 BezierBasisPoint

BezierBasisPoint returns the single value of single parameter in the basis function.

Usage

BezierBasisPoint(Upper,Lower,Independent)

Parameters

Upper - the degree (n) of the Bernstein basis function.

Lower - i , the required coefficient of the Bernstein basis function.

Independent - the value for which the Bernstein basis coefficient is required.

Remarks

The function returns a single coefficient of the BernsteinBasis polynomial:

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad (3)$$

with

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \quad (4)$$

6 BezierCoefficient

BezierCoefficient calculates the Bernstein basis factorial function coefficient.

Usage

BezierCoefficient(Upper,Lower)

Parameters

Upper - the degree (n) of the Bernstein basis function.

Lower - i , the required coefficient of the Bernstein basis function.

Remarks

The function returns a single factorial function coefficient of the Bernstein-Basis polynomial:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \quad (5)$$

7 BezierCurve

BezierCurve allows the construction of a Bezier spline curve of user supplied degree.

Usage

BezierCurve(*ControlVertices, CurvePoints, Degree*)

Parameters

ControlVertices - a three column by n row array of control vertices.

CurvePoints - a single column by m row array of required interpolation points.

Degree - the required degree of the basis polynomial.

Remarks

A BezierCurve is a special case of a non-uniform rational basis spline. Its construction is determined by its control polygon and its basis is the Bernstein basis. Bernstein basis functions are real, with their degree being one less than the number of polygon control points. The first and last points on the curve form the first and last points of the control polygon. The BezierCurve is fully contained within the convex hull of its control polygon and is invariant under an affine transformation.

The function returns an array of 2 columns by m rows of interpolated points.

8 BezierCurveDerivatives

BezierCurveDerivatives calculates the first and second derivatives of a Bezier spline curve of user supplied degree.

Usage

BezierCurve(*CurvePoints, Degree, FirstOrSecond*)

Parameters

CurvePoints - a single column by m row array of required interpolation points.

Degree - the required degree of the basis polynomial.

FirstOrSecond - an integer of value either 1 or 2 to specify the first or second derivatives as required.

Remarks

Specification of the numerical tangent vectors at the ends of a BezierCurve is not strictly necessary, though it can provide useful information when trying to maintain slope or continuity, or when joining successive BezierCurves. The function returns an array of m rows by degree -1 columns of either first or second derivatives.

BernsteinBasis

9 CatMullRomSpline

Interpolate using a Catmull-Rom recursive basis-spline.

Usage

CatmullRomSpline(*Independent, Dependent,*
Required, NumberOfKnots)

Parameters

Independent - a contiguous single column array of monotonically increasing values. Must have the same number of observations as the *Dependent* array.

Dependent - a contiguous single column array of values. Must have the same number of values as the *Independent* array.

Required - contiguous single column array containing one or more independent values for which the splined dependent value is required. The values must be ordered in same direction as the independent values.

NumberOfKnots - a single integer value fixing the number of knots or control points used to calculate the spline. Permissible values are 2 or 4.

Remarks

Catmull-Rom splines can be used either as interpolating or as approximating splines. If used as interpolating splines, the piece-wise cubic polynomial segments pass through all the control points except the first and last. It should be noted that Catmull-Rom splines do not possess the convex hull property (see “An Introduction to the Mathematics and Construction of Splines” for details).

10 GSpline

Interpolate using splines constructed using a factorisation method.

Usage

GSpline(*Independent, Dependent, Weights,*
Derivatives, Degree, Required)

Parameters

Independent - a contiguous single column array of monotonically increasing values. Must have the same number of observations as the *Dependent* array.

Dependent - a contiguous single column array of values. Must have the same number of values as the *Independent* array.

Weights - a contiguous single column array of the integer weighting values to be applied to each of the *Independent* values. The weights indicate the number of observations at each point of the *Independent* array. None of the values of the *Weights* array should exceed the value of the *Degree* of the spline.

Derivatives - is a matrix of derivatives. The number of rows should be equal to the number of observations and the number of columns should be the same as the *Degree* of the spline.

Degree - a positive integer indicating the desired degree of the spline to be fitted to the supplied *Independent* and *Dependent* data.

Required - contiguous single column array containing one or more independent values for which the splined dependent value is required. The values must be ordered in same direction as the independent values.

Remarks

This function evaluates an interpolatory spline for Hermite-Birkoff data using a factorisation method developed by Eidson and Schumaker. In order for

the G-spline function to provide stable and reliable results, the number of observations must be greater than or equal to the degree of the requested spline fit. If the function fails to find a fit, it will return an error message informing the user of the reason for its failure. This will in general occur if a singular system is encountered in the implied spline polynomial structure.

11 MultipleOpenKnotVector

MultipleOpenKnotVector generates a basis spline open knot vector with multiplicity equal to the order at the end points.

Usage

MultipleOpenKnotVector(*NumberOfVertices*, *Order*)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

Remarks

The function returns a vector containing the knot vector.

12 NonUniformOpenKnotVector

NonUniformOpenKnotVector generates a non-uniform open knot vector proportional to the chord lengths between defining polygon vertices.

Usage

NonUniformOpenKnotVector(*NumberOfVertices*, *Order*, *Vertices*)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

Vertices - array of polygon vertices.

Remarks

Non-uniform basis-spline curves are generated using a knot vector with interior knot values proportional to the chord distance between the polygon vertices, with the knot vector specified by the following equations:

$$x_i = 0 \quad 1 \leq i \leq k \quad (6)$$

$$x_{i+k} = \frac{\binom{i}{n-k+2} c_{i+1} + \sum_{j=1}^i c_j}{\sum_{i=1}^n c_i} (n-k+2) \quad 1 \leq i \leq n-k+1 \quad (7)$$

$$x_i = n-k+2 \quad n+1 \leq i \leq n+k \quad (8)$$

13 OpenKnotBasisSplineVector

OpenKnotBasisSplineVector calculates basis functions for open knot vectors.

Usage

OpenKnotBasisSplineVector(*NumberOfVertices*, *Order*, *Knots*, *Independent*)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

Knots - vector of knot points

Independent - value for which basis spline coefficient is required.

Remarks

The knot vector must be a monotonically increasing series of real numbers.

14 OpenKnotVector

OpenKnotVector calculates a basis spline open knot vector.

Usage

OpenKnotVector(*NumberOfVertices*, *Order*)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

Remarks

For a basis spline curve of order n (degree = $n - 1$), a point on the curve will lie within the convex hull of n neighbouring points.

15 OpenUniformKnotBasisSplineCurve

OpenUniformKnotBasisSplineCurve calculates a basis spline curve, given an open uniform knot vector.

Usage

OpenUniformKnotBasisSplineCurve(*NumberOfVertices*, *Order*, *NumberOfPoints*, *Vertices*)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

NumberOfPoints - number of points to be calculated on the curve.

Vertices - array of polygon vertices.

Remarks

The supplied knot values are assumed to be evenly spaced.

16 OpenUniformKnotRBasisSplineCurve

OpenUniformKnotRBasisSplineCurve calculates a rational basis spline curve, given an open uniform knot vector.

Usage

OpenUniformKnotRBasisSplineCurve(*NumberOfVertices*, *Order*, *NumberOfPoints*, *Vertices*, *Weights*)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

NumberOfPoints - number of points to be calculated on the curve.

Vertices - array of polygon vertices.

Weights - array of weights to be applied in the construction of the basis spline.

Remarks

The supplied knot values are assumed to be evenly spaced for the construction of the rational basis spline to complete correctly.

17 QuinticSpline

Interpolate using quintic natural splines

Usage

QuinticSpline(*Independent, Dependent, Required, Derivatives*)

Parameters

Independent - a contiguous single column array of monotonically increasing values. Must have the same number of observations as the *Dependent* array.

Dependent - a contiguous single column array of values. Must have the same number of values as the *Independent* array.

Required - contiguous single column array containing one or more independent values for which the splined dependent value is required. The values must be ordered in same direction as the independent values.

Derivatives - this is a boolean variable. If set to "True", the function returns an array with the same number of rows as the *Independent* and *Dependent* arrays and with five columns - one for each of the derivatives of the quintic basis function.

Remarks

The calculation of the coefficients if the spline function uses minimum support B-splines of degree 2 to form a basis for the class of third derivatives of the quintic natural splines. The knot points provided in the **Independent** array are generally preferred to be monotonically increasing, but the function will support both double and triple knots

18 RationalBasisSplineOpenKnotVector

RationalBasisSplineOpenKnotVector calculates rational basis spline functions using an open knot vector.

Usage

RationalBasisSplineOpenKnotVector(*NumberOfVertices, Order, Independent, Knots, Weights*)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

Independent - value for which basis spline coefficient is required.

Knots - vector of n knot points

Weights - array of n weights to be applied in the construction of the basis spline.

Remarks

The *Knots* and *Weights* vectors must be the same length or the function will return an error.

19 TensionSpline

Interpolate using splines under constant uniform tension.

Usage

TensionSpline(*Independent, Dependent, Required,*
[Curve, SlopeStart, SlopeEnd, Tension])

Parameters

Independent - a contiguous single column array of monotonically increasing values. Must have the same number of observations as the *Dependent* array.

Dependent - a contiguous single column array of values. Must have the same number of values as the *Independent* array.

Required - contiguous single column array containing one or more independent values for which the splined dependent value is required. The values must be ordered in same direction as the independent values.

Curve - a boolean variable. Set to TRUE to return a merged array containing both the original and splined values. Set to FALSE to return only splined values for the required points.

SlopeStart - value for the slope at the beginning of the data series. If *SlopeStart* is set to zero, the function will automatically determine the value.

SlopeEnd - value of the slope at the beginning of the data series. If set to zero the function will automatically determine the value.

Tension - a strictly positive value for the tension to be applied to the construction of the curve. The lower limit is 0 and the upper limit is 100. Higher values will produce a more “curvy” and smoother fit, whilst values nearer to 0 will generate a fit that will become linear when *Tension* reaches 0.

Remarks

The curve is a spline under tension, which is somewhat “tighter” than a cubic spline and less likely to have spurious inflection points. If the *Curve* variable is set to TRUE, the entire curve will be returned with the given points included in the output. An explanation of the mathematical theory of splines under tension can be found in the reference document which forms part of this product.

20 UniformOpenBasisSplineCurve

UniformOpenBasisSplineCurve calculates a basis spline curve using a uniform open knot vector.

Usage

UniformOpenBasisSplineCurve(*NumberOfVertices, Order, Vertices,*
NumberOfPoints, NumberOfVertices)

Parameters

NumberOfVertices - the number of defining polygon vertices.

Order - order of the basis function.

Vertices - array of polygon vertices.

NumberOfPoints - number of points to interpolate.

NumberOfVertices - number of control vertices.

Remarks

As the name, suggests, the knot points must be evenly spaced. In practice, the generally adopted approach is to normalise the knot values such that the knot values are in the range 0 to 1.

21 UniformPeriodicKnotVector

UniformPeriodicKnotVector calculates a basis spline vector given an array of uniformly spaced periodic knots.

Usage

UniformPeriodicKnotVector(*NumberOfVertices*, *Order*)

Parameters

NumberOfVertices - number of control vertices.

Order - order of the basis function.

Remarks

Two types of knot vector can be used: periodic and open. Each type can be used in either uniform or non-uniform format. In the case of uniform periodic knot vectors, it is generally accepted practice to begin knot vectors at zero and normalise the values to lie in the range 0 to 1.

22 X-Spline

Interpolate using splines that can be either interpolating or approximating.

Usage

XSpline(*FirstDataSet*, *SecondDataSet*, *ThirdDataSet*,
Weights, *Knots*, *Segments*, *Delta*, *Increment*)

Parameters

FirstDataSet - a contiguous single column array of monotonically increasing values. Must have the same number of observations as the SeondDataSet and ThirdDataSet arrays.

SecondDataSet - a contiguous single column array of values. Must have the same number of as the FirstDataSet and ThirdDataSet arrays.

ThirdDataSet - a contiguous single column array of values. Must have the same number of as the FirstDataSet and SeondDataSet arrays.

Weights - a contiguous single column array of weighting values. All values must be in the range [0, 1] Must have the same number of values as the FirstDataSet array.

Knots - the number of control points or knots to be used to calculate the splined curve.

Segments - the number of sub-divisions of the curve. Calculated as Knots x Delta.

Delta - the distance between each knot point.

Increment - the size of the step between Knots.

Remarks

The curve is an x-spline which provides the ability to either interpolate or approximate the supplied data. An explanation of the mathematical theory of x-splines can be found in the reference document which forms part of this product.

Addix Financial Software

September 2002

Support via the net: support@addix.com

Telephone: 07020-939710