Shape Parametrisation Utilities for the Optimisation Shell INVERSE

(FOR VERSION 3.11)

Igor Grešovnik

Ljubljana, 27 September, 2005

Contents:

9. SHAPE PARAMETRISATION UTILITIES

9.1 Cubis Splines with Prescribed Endpoint Derivatives

9.1.1 About Splines

Spline interpolation is a piecewise polynomial interpolation (Figure 9.1). An interval $[a,b]$ on which we want to interpolate the function $f(x)$ by a spline $g(x)$ is subdivided into $n-1$ subintervals with common endpoints called nodes:

$$a = x_1 < x_2 < x_3 < \ ... \ < x_n = b \qquad (9.1)$$

We require that functions *f* and *g* match on nodes, i.e.

$$f(x_i) = g(x_i), \ i = 1, 2, ..., n. \qquad (9.2)$$

We require that function $g(x)$ is a polynomial of a certain degree on each subinterval, i.e.

$$g(x) = \left\{ \begin{matrix} p_1(x); \ x \le x_2 \\ p_2(x); \ x_2 \in [x_2, x_3] \\ ... \\ p_{n-1}(x); \ x \ge x_{n-1} \end{matrix} \right\} \qquad (9.3)$$

We also require certain continuity conditions to be fulfilled in nodes (e.g. continuity of the first and second derivatives) and appropriate boundary conditions to be fulfilled in the endpoints *a* and *b* (e.g. we specify values of the first derivatives in these points). Points (x_i, y_i), where x_i is the *i*-th node and y_i is the prescribed value of the spline in this node, are called **control points**.

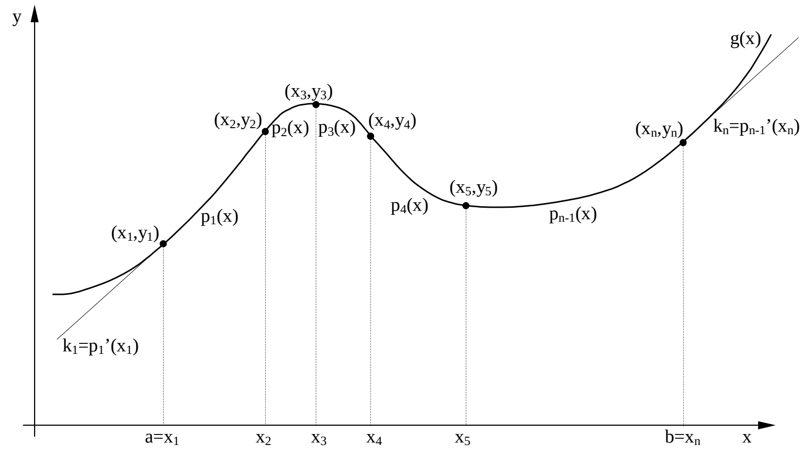

Figure 9.1: Cubic spline with 6 control points.

9.1.1.1 Cubic Splines with Prescribed Endpoint Derivatives

A cubic spline $g(x)$ is a function of the form (9.3), which is represented in each subinterval of the subdivision (9.1) by a **polynomial of the third or lesser degree**, and is a **continuous** function with **continuous first and second derivatives** in the interval $[a,b]$.

A cubic polynomial in any interval of the subdivision (9.1) is uniquely determined by endpoint values and derivatives:

$$\begin{aligned} p_j(x) &= y_j d_j^2 (x-x_{j+1})^2 (1+2d_j(x-x_j)) \\ &+ y_{j+1} d_j^2 (x-x_j)^2 (1-2d_j(x-x_{j+1})) \\ &+ k_j d_j^2 (x-x_j)(x-x_{j+1})^2 + k_{j+1} d_j^2 (x-x_j)^2 (x-x_{j+1}) \end{aligned} , \qquad (9.4)$$

where

$$p_j(x_j) = f(x_j) = y_j,\; p_j(x_{j+1}) = f(x_{j+1}) = y_{j+1},\; j = 1,\ldots,n \;, \qquad (9.5)$$

$$d_j = \frac{1}{x_{j+1} - x_j} \qquad (9.6)$$

and

$$k_j = p_j'(x_j). \qquad (9.7)$$

k_1 and k_n must be prescribed while other derivatives k_i are determined by evaluation of the linear system of equations which is obtained by taking into account the continuity and boundary conditions, namely

$$\mathbf{A}_{(n-2)\times(n-2)}\mathbf{u}_{(n-2)} = \mathbf{b}_{(n-2)} \quad , \tag{9.8}$$

where

$$u_i = k_{i+1},\ i = 1, 2, ..., n-2 \,. \tag{9.9}$$

Since computation of spline values involves solution of the system (9.8) (which can be solved only once prior to any evaluation of $g(x)$ or $p_j(x)$, respectively), derivatives with respect to any parameter ϕ involve implicit terms $\frac{dk_i}{d\phi} = \frac{du_{i-1}}{d\phi}$:

$$\frac{dp_j(x)}{d\phi} = \frac{\partial p_j(x)}{\partial \phi} + \sum_{i=2}^{n-1} \frac{\partial p_j(x)}{\partial k_i}\frac{dk_i}{d\phi} \,. \tag{9.10}$$

These terms are obtained by derivation of (9.8) with respect to the parameter ϕ_i:

$$\mathbf{A}\frac{d\mathbf{u}}{d\phi} = \frac{d\mathbf{b}}{d\phi} - \frac{d\mathbf{A}}{d\phi}\mathbf{u} \,. \tag{9.11}$$

ϕ can be any parameter that defines the spline, i.e. x or y coordinate of any node (i.e. x_i or y_i) or the prescribed derivative in the first or the last node (i.e. k_1 or k_n). The derivative of the vector of derivatives of the spline in internal nodes ($d\mathbf{u}/d\phi$) can be evaluated prior to evaluation of $\partial p_j(x)/d\phi$ in any point x. Moreover, the linear system for evaluation of $d\mathbf{u}/d\phi$ has the same system matrix as the system for evaluation of $\mathbf{u}$ itself, therefore the factorized matrix that was calculated to solve the system for $\mathbf{u}$ can be reused.

9.2 Shell Built-in Splines Utilities

The optimisation shell *Inverse* includes a set of basic utilities for doing spline calculation. User can define splines by specifying control points and endpoint derivatives, calculate values on defined splines and derivatives of these values with respect to any parameter of a spline (x or y coordinate of individual control points or endpoint derivatives). Calculator functions enable both defining splines and performing calculations on them, but it is better to use file interpreter functions for defining splines because in this case x and y coordinates of all control points defining the splines are gathered in vectors.

More than one spline can be defined at once. Different splines are referred to by identification numbers. There is however one spline, referred to in this text as the “global spline”, which needs not to be referred by an identification number. When using functions that require the spline identification number, this spline can be referred to by the identification number 0. The global spline is introduced for the sake of simplicity; the user is in this way not forced to use and keep trace of spline identification numbers if it is sufficient for a specific problem that only one spline is defined at once.

9.3 File Interpreter Functions for Manipulating Splines

File interpreter functions enable definition of splines and their destruction.

9.3.1 setnewcubspl *{ x y k1 kn idvar }*

Sets a new cubic spline defined by x coordinates of control points *x*, y coordinates of control points *y* (both vector arguments) and prescribed derivatives in the first and the last control points, *k1* and *kn* (value arguments), respectively. The function assigns the identification number of the newly defined spline to a calculator variable named *idvar*. The identification number of the newly defined spline is determined by the spline system and must be specified any time when this spline is referred after its definition.

9.3.2 setcubsplid *{ id x y k1 kn }*

Defines a cubic spline that has identification number *id* (value argument). If the spline with identification number *id* is already defined, its definition is overridden by the new one. If it does not exist yet, a new spline is created in the spline system and identification number *id* is assigned to it. In this case *id* should not be too large because spline identification numbers are internally stored in a table that has as many elements as is the largest currently existing identification number. *id* must in any case evaluate to an integer greater than or equal to 0. If *id* is 0, the definition affects the global spline.

The spline gets defined by x coordinates of control points *x*, y coordinates of control points *y* (both vector arguments) and prescribed derivatives in the first and the last control points, *k1* and *kn* (value arguments), respectively.

9.3.3 setcubsplglob *{ x y k1 kn }*

Defines the global cubic spline by x coordinates of control points *x*, y coordinates of control points *y* (both vector arguments) and prescribed derivatives in the first and the last control points, *k1* and *kn* (value arguments), respectively.

9.3.4 dispcubspl *{ <id> }*

Deletes the definition of the cubic spline identified by *id* (value argument) togerher with associated internal auxiliary data structures. Identification numver *id* is available for new splines (e.g. defined by the *setnewcubspl* interpreter function) after this function is executed. If *id* is not specified, the function deletes the global cubic spline.

9.3.5 fprintcubspldata *{ <id> }*

Prints the data about splines defined in the cubic spline system to the output file of the programme (i.e. the file *outfile*) . If *id* is specified, only the data about the cubic spline with identification number *id* is printed.

9.3.6 printcubspldata *{ <id> }*

Prints the data about splines defined in the cubic spline system to the standard output of the programme. If *id* is specified, only the data about the cubic spline with identification number *id* is printed.

9.4 Expression Evaluator Functions for Manipulating Splines

9.4.1 setnewcubspl *[k1, kn, x1, y1, x2, y2, …]*

Defines a new cubic spline and returns its identification number. Arguments *k1* and *kn* are the prescribed derivatives in the first and the last control points, *x1* and *y1* are x and y coordinates of the first control point, *x2* and *y2* are coordinates of the second control point, etc. There must be at least two control points (at minimum 6 arguments) and the number of arguments must be even.

Remark:

It is more common to use the interpreter function with the same name instead of this calculator function. x and y coordinates of control points are specified by vector arguments when using the interpreter function.

9.4.2 setcubspl *[<id>, k1, kn, x1, y1, x2, y2, …]*

Defines the cubic spline identified by *id*. If the spline with this identification number already exists, its definition is overridden. If it does not exist, it is created anew, in which case *id* should not be too large because spline identification numbers are internally stored in a table that has as many elements as is the largest currently existing identification number. *id* must evaluate to an integer greater than or equal to 0. If *id* is 0 or it is not specified, the global cubic spline is defined.

Arguments *k1* and *kn* are the prescribed derivatives in the first and the last control points, *x1* and *y1* are x and y coordinates of the first control point, *x2* and *y2* are coordinates of the second control point, etc. There must be at least two control points (at minimum 6 arguments).

Remark:

It is more common to use the interpreter function *setcubsplid* or *setcubsplglob* instead of this calculator function. x and y coordinates of control points are specified by vector arguments when using the interpreter function.

9.4.3 dispcubspl *[<id>]*

Deletes the definition of the cubic spline identified by *id*. If *id* is not specified, this affects the global cubic spline.

Remark:

It is more common to use the interpreter function with the same name instead of this calculator function.

9.4.4 cubspl *[<id>, x]* , cubsplval

Evaluates to the value of the cubic spline with identification number *id* at *x*. If *id* is not specified or it is zero, the value of the global cubic spline is calculated.

9.4.5 cubsplsensy *[<id>, i, x]*

Evaluates to sensitivity of the value of the cubic spline with identification number *id* at *x* with respect to the y coordinate of the *i*-th control point that defines the spline. If *id* is not specified or it is zero, the function refers to the global spline.

9.4.6 cubsplsensx *[<id>, i, x]*

Evaluates to the sensitivity of the value of the cubic spline with identification number *id* at *x* with respect to the x coordinate of the *i*-th control point that defines the spline. If *id* is not specified or it is zero, the function refers to the global spline.

9.4.7 cubsplsensk1 *[<id>, x]*

Evaluates to the sensitivity of the value of the cubic spline with identification number *id* at *x* with respect to the prescribed derivative in the first node. If *id* is not specified or it evaluates to zero, the function refers to the global spline.

9.4.8 cubsplsenskn *[<id>, x]*

Evaluates to the sensitivity of the value of the cubic spline with identification number *id* at *x* with respect to the prescribed derivative in the last node. If *id* is not specified or it evaluates to zero, the function refers to the global spline.