

# 1 Constraint Based Curve Manipulation

In this section we consider taking a curve, subjecting it to some requirement for modification, imposing some constraints, and then adjusting the curve so that it responds appropriately. For spline curves, such adjustments may involve changing knots, control vertices, and/or weights (if the curve is rational). We shall be restricting ourselves later in this section to linear adjustments and constraints. Since spline curves depend linearly only on their control vertices, any consideration of knot and weight adjustments will be dropped.

As a simple example, we might require that an existing curve pass through a designated point and investigate what adjustments of the control vertices would meet this requirement. Since a typical curve will depend upon  $m + 1$  control vertices,  $P_0, \dots, P_m$ , and the example imposes only one requirement, we might expect that there are something like  $m$  degrees of freedom left unspecified, providing no unique solution to the problem. In such a case, we shall resolve the ambiguity by requiring the least possible adjustment to the curve's control vertices in some metric.

To be more specific, let

$$\sum_{i=0}^m P_i N_i^n(u) \quad \text{and} \quad \sum_{i=0}^m [P_i + d_i] N_i^n(u)$$

represent the curve in its initial position and after modification, respectively. Modifications are specified by requiring a *target functional*,  $\phi$ ; e.g. to achieve a *specified value* or a *minimum*

$$\phi \left( \sum_{i=0}^m [P_i + d_i] N_i^n(u) \right) = \lambda \quad \text{or} \quad \text{minimize } \phi \left( \sum_{i=0}^m [P_i + d_i] N_i^n(u) \right)$$

In dynamic situations,  $\phi$  and  $\lambda$  may be functions of time.

Constraints are specified by *constraint functionals*  $f_j$ ,  $j = 0, \dots, c$  and *constraint values*  $\gamma_j$ . For each  $j$  the constraint may be an *equality constraint* or an *inequality constraint*; that is, respectively,

$$f_j \left( \sum_{i=0}^m [P_i + d_i] N_i^n(u) \right) = \gamma_j \quad \text{or} \quad f_j \left( \sum_{i=0}^m [P_i + d_i] N_i^n(u) \right) \geq \gamma_j$$

Again, these constraints may depend upon time.

A *least adjustment* will be given by a metric  $\mu$  and the requirement that  $\mu(d_0, \dots, d_m) = \min$ .

Thus, in general, we must confront a value-finding or minimization problem involving  $\phi$ , subject to  $c + 1$  constraints in the  $m + 1$  variables  $d_i$  with the possible side condition defined by  $\mu$ . To see how target and constraint functionals might be derived, consult [17], where functionals corresponding to a number of physically meaningful situations are presented. [2] catalogs a more specific list of geometric constraints defined in terms of functionals involving the control points and weights of a rational Bézier curve. In [13] the target functional imposes “fairness” on a planar curve and a functional constraining area is added. For surfaces, [12] employs a fairness functional as the target; constraints are used to enforce continuity between patches and provide certain shape characteristics. A reference for which the target functional is defined by a data approximation problem, and the constraints are simple requirements on the control vertices, is to be found in [14]. Going to extremes, one may use a target functional alone with no constraints [10]. A good overview of numerical approaches to solving constrained problems, when solutions exist, is given in [8]. A useful reference for the numerical approaches to unconstrained problems would be [5].

We cannot assume that a solution exists for arbitrary versions of the problem; in the references above, the functionals were chosen with care. For restricted settings, however, assurances of a solution will be available. In [4, 16], for example, positive-definite quadratic  $\phi$ 's are considered, and the  $f_j$  are linear equality constraints, which reduces the problem to a simple form of quadratic programming.

In this section we shall restrict the generality to a further degree by requiring that  $\phi$  be linear and attain a specified value, that only linear equality constraints be present, that  $\mu$  be

$$\mu(d_0, \dots, d_m) = \left( \sum_{i=0}^m \|d_i\|^2 \right)^{\frac{1}{2}} = \min \quad (1)$$

where  $\|\cdot\|$  represents the ordinary Euclidian norm, and that there be no more functionals than control vertices. With these restrictions, and with the assumption that the functionals are all linearly independent, a unique set of  $d_i$  exists. In this setting,  $\phi$  need not be distinguished in any way from the  $f_j$ , since all are linear and each is required to attain a specified value.

Thus, we shall consider finding  $d_0, \dots, d_m$  so that (1) and

$$f_j \left( \sum_{i=0}^m [P_i + d_i] N_i^n(u) \right) = \gamma_j \quad (2)$$

holds for  $j = 0, \dots, c$ . In this context, the issue of whether any functional depends on time is irrelevant. Time can be handled as a sequence of discrete events, treating each event as a static problem to be solved. This is implicit for the interactive settings described in [1, 6, 7]. The treatment we present here will follow the material in these three references, but it has a much wider applicability. For example in [3], target functionals provide physically-based forms of dynamic modification, and linear constraints, presented there with a geometric interpretation, are treated essentially as in [7]. In [11], precisely the same treatment of linear constraints as in [7] is applied to free-form deformations to achieve a means of direct interactive manipulation on surfaces. To review the linear algebra we employ below, consult [9].

Since the  $f_j$  are linear, the  $j^{\text{th}}$  equation of (2) can be rewritten as

$$\sum_{i=0}^m F_{ji} d_i = \beta_j, \quad \text{where } F_{ji} = f_j(N_i^n(u)) \quad \text{and} \quad \beta_j = \gamma_j - \sum_{i=0}^m P_i f_j(N_i^n(u))$$

or, more compactly in matrix-vector notation

$$\mathbf{F} \mathbf{d} = \mathbf{b} \quad (3)$$

Since we have no more functionals than control vertices,  $\mathbf{F}$  will be square, or it will have fewer rows than columns (the usual case). The linear independence of the  $f_j$  implies that the rows of  $\mathbf{F}$  will be linearly independent. A common special case is the one in which some of the  $f_j$  are chosen to *preserve* a geometric property. That is, some of the  $f_j$  already take on the value  $\gamma_j$  for the original, unmodified curve, and correspondingly  $\beta_j = 0$ .

To find a solution to (3) with side condition (1), we write  $\mathbf{d}$  in terms of *mutually orthogonal* components, one in the *row space* of  $\mathbf{F}$  and the other in the *null space* of  $\mathbf{F}$  [15]:

$$\mathbf{d} = \mathbf{F}^T \mathbf{v} + \mathbf{z} \quad (4)$$

where  $\mathbf{v}$  is some vector of length  $c + 1$ , and  $\mathbf{F} \mathbf{z} = 0$  (implying that the dot product of  $(\mathbf{F}^T \mathbf{v})$  with  $\mathbf{z}$  is 0). With this representation, we see that

$$\|\mathbf{d}\|^2 = (\mathbf{F}^T \mathbf{v} + \mathbf{z}) \cdot (\mathbf{F}^T \mathbf{v} + \mathbf{z}) = (\mathbf{F}^T \mathbf{v}) \cdot (\mathbf{F}^T \mathbf{v}) + \mathbf{z} \cdot \mathbf{z} = \|\mathbf{F}^T \mathbf{v}\|^2 + \|\mathbf{z}\|^2$$

We wish to solve (3) with a vector  $\mathbf{d}$  of minimum length. We begin by observing that this system does not depend on  $\mathbf{z}$ :

$$\mathbf{b} = \mathbf{F}\mathbf{d} = \mathbf{F}(\mathbf{F}^T\mathbf{v} + \mathbf{z}) = \mathbf{F}\mathbf{F}^T\mathbf{v} + \mathbf{F}\mathbf{z} = \mathbf{F}\mathbf{F}^T\mathbf{v}$$

and therefore

$$\mathbf{v} = (\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{b} \quad (5)$$

The components of  $\mathbf{z}$  constitute parameters irrelevant to the system, yet they contribute to the length of  $\mathbf{d}$ , so we obtain the minimum length solution to (3) when by setting  $\mathbf{z} = 0$ . Substituting (5) into (4) with this setting leads to

$$\mathbf{d} = \mathbf{F}^T(\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{b}$$

where the matrix  $\mathbf{F}\mathbf{F}^T$  is nonsingular under our assumption that the rows of  $\mathbf{F}$  are independent.

Since the inverse of  $\hat{\mathbf{F}} = \mathbf{F}\mathbf{F}^T$  is given by  $\text{adj}(\hat{\mathbf{F}})/\det(\hat{\mathbf{F}})$ , where  $\text{adj}(\hat{\mathbf{F}})$  is the adjoint matrix of  $\hat{\mathbf{F}}$ , this solution can be re-expressed as

$$\mathbf{d} = \mathbf{F}^T \text{adj}(\hat{\mathbf{F}})\mathbf{b}/\hat{f} \quad (6)$$

where  $\hat{f} = \det(\hat{\mathbf{F}})$ . Letting  $\mathbf{F}_i$  and  $\mathbf{F}_j$  denote the  $i^{\text{th}}$  and  $j^{\text{th}}$  rows of  $\mathbf{F}$ , the  $i, j^{\text{th}}$  element of  $\hat{\mathbf{F}}$  is  $\hat{f}_{i,j} = \mathbf{F}_i \cdot \mathbf{F}_j$ . The cofactor constituting the  $i, j^{\text{th}}$  element of  $\text{adj}(\hat{\mathbf{F}})$  will be denoted by  $\alpha_{i,j}$ , and for small (i.e. three rows or less) systems, these cofactors may be expressed compactly in terms of the elements of  $\hat{\mathbf{F}}$ . With large numbers of constraints it would be inefficient to compute  $\mathbf{d}$  using (6), and a *QR decomposition* of  $\mathbf{F}^T$  should be used instead [9]. However, for small sets of constraints, particularly in an interactive setting, this is a very good approach to take.

If these constraints are to be applied during interactive manipulation, we divide the solution into a *preprocessing* stage and an *iterative* stage. In the preprocessing stage we compute  $\mathbf{C} = \mathbf{F}^T \text{adj}(\hat{\mathbf{F}})/\hat{f}$ . The iterative stage consists merely of multiplying a sequence of  $\mathbf{b}$ 's by  $\mathbf{C}$  to obtain a sequence of  $\mathbf{d}$ 's. Only those columns of  $\mathbf{C}$  corresponding to nonzero components of  $\mathbf{b}$  need be computed. The preprocessing stage is performed at the beginning of an interaction; e.g. at the press of a mouse button, and the iterative stage is applied throughout the interaction; e.g. during movement of the mouse.

A system composed of a single constraint is given by

$$\mathbf{F}_0\mathbf{d} = \beta_0 \quad (7)$$

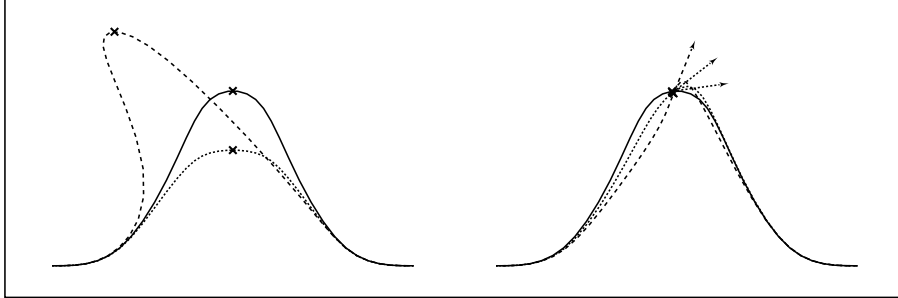


Figure 1: A single constraint.

The solution for this system is

$$\mathbf{d} = \mathbf{F}_0^T \beta_0 / \hat{f}_{0,0}$$

At a given point  $\bar{u}$  on the curve, (7) can be used to constrain position, tangency, or a higher derivative by setting the  $i^{\text{th}}$  component of  $\mathbf{F}_0$  equal to the value or appropriate derivative of  $N_i^n(u)$  evaluated at  $\bar{u}$ . However, the position of the curve at  $\bar{u}$  will vary if any derivative is constrained. Figure 1 illustrates two examples. It shows the application of a single constraint at “ $\times$ ”. The original curve is solid, while successive manipulations are dotted and dashed. In (a), a positional constraint is applied while higher derivatives are free to vary. In (b), the tangent is changed.

By constraining the position at  $\bar{u}$ , while applying change through properties expressible in terms one or more other functionals, modifications may be controlled directly and intuitively at the fixed position on the curve. Two such examples are special cases of manipulations that impose two, respectively three, constraints.

For a system of two constraints:

$$\begin{bmatrix} \mathbf{F}_0 \\ \mathbf{F}_1 \end{bmatrix} \mathbf{d} = \begin{bmatrix} 0 \\ \beta_1 \end{bmatrix}$$

whose solution is:

$$\mathbf{d} = \mathbf{F}^T \begin{bmatrix} -\hat{f}_{0,1} & \hat{f}_{0,0} \end{bmatrix}^T \beta_1 / \hat{f}$$

where

$$\hat{f} = \hat{f}_{0,0} \hat{f}_{1,1} - \hat{f}_{0,1}^2$$

The position of the curve is fixed by setting the components of  $\mathbf{F}_0$  equal to the values of the functions  $N_i^n(u)$  at a point  $\bar{u}$ . Examples of such systems are given in Figure 2. The figure shows the application of a double constraint at “ $\times$ ”. Tangent direction and magnitude are varied in (a) and (b), respectively, while position is fixed.

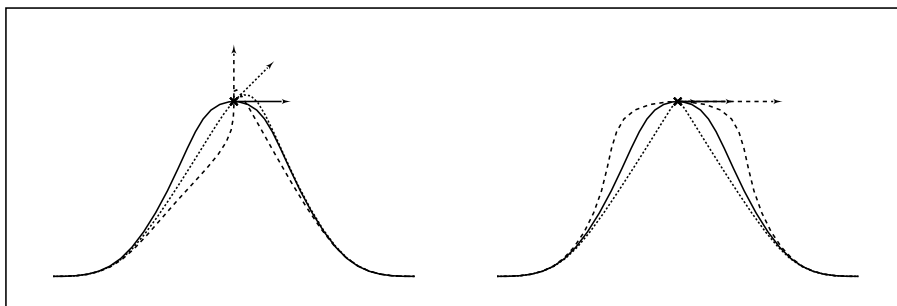


Figure 2: A double constraint.

Three constraints can be applied using a system of the form

$$\begin{bmatrix} \mathbf{F}_0 \\ \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ \beta_2 \end{bmatrix}$$

the solution to which is given by

$$\mathbf{d} = \mathbf{F}^T \begin{bmatrix} \alpha_{0,2} & \alpha_{1,2} & \alpha_{2,2} \end{bmatrix}^T \beta_2 / \hat{f}$$

where

$$\begin{aligned} \alpha_{0,2} &= \hat{f}_{0,1}\hat{f}_{1,2} - \hat{f}_{1,1}\hat{f}_{0,2} \\ \alpha_{1,2} &= \hat{f}_{0,1}\hat{f}_{0,2} - \hat{f}_{0,0}\hat{f}_{1,2} \\ \alpha_{2,2} &= \hat{f}_{0,0}\hat{f}_{1,1} - \hat{f}_{0,1}^2 \end{aligned}$$

and

$$\hat{f} = \hat{f}_{0,2}\alpha_{0,2} + \hat{f}_{1,2}\alpha_{1,2} + \hat{f}_{2,2}\alpha_{2,2}.$$

Figure 3 illustrates two applications of the triple constraint system. In (a), tangency is varied while preserving position and second derivative. In (b), position is varied while curvature (determined by first and second derivatives) is preserved.

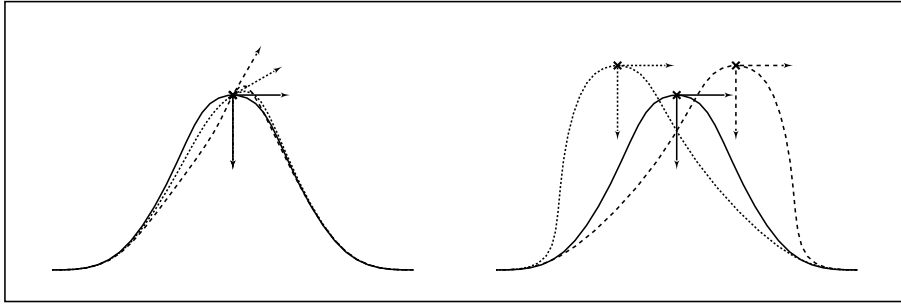


Figure 3: A triple constraint.

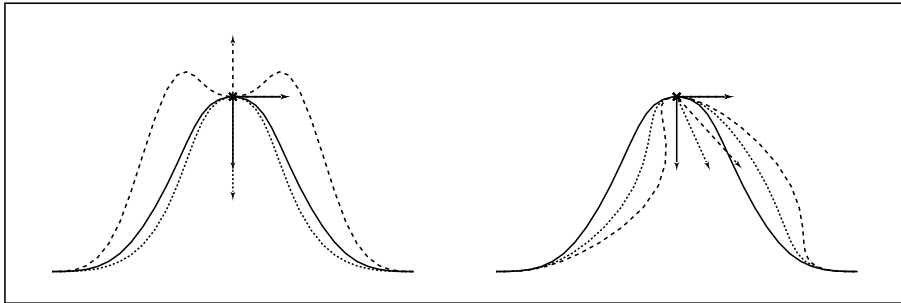


Figure 4: Modifying curvature.

Figure 4 illustrates manipulations of the second derivative vector relative to a fixed position and first derivative vector that alter and preserve curvature. In (a), curvature magnitude is varied by moving the second derivative vector *perpendicular* to the first derivative vector. In (b), curvature magnitude is preserved by moving the second derivative vector *parallel* to the first derivative vector.

## References

- [1] Richard H. Bartels and John C. Beatty. A technique for the direct manipulation of spline curves. In *Graphics Interface '89*, pages 33–39, London, Ontario, 1989. Morgan Kaufmann Publishers, Palo Alto, California.
- [2] Paula L. Beaty, Patrick A. Fitzhorn, and Gary J. Herron. Extensions in variational geometry that generate and modify object edges composed of rational bézier curves. *Computer-Aided Design*, 26(2):98–108, February 1994.
- [3] George Celniker and Dave Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics*, 25(4):257–266, 1991. SIGGRAPH '91 conference proceedings.
- [4] George Celniker and William Welch. Linear constraints for deformable non-uniform B-spline surfaces. In *1992 Symposium on Interactive 3D Graphics*, pages 165–170, Cambridge, Massachusetts, 1992. Special issue of *Computer Graphics*.
- [5] John E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, 1983.
- [6] Barry Fowler. Geometric manipulation of tensor product surfaces. In *1992 Symposium on Interactive 3D Graphics*, pages 101–108, Cambridge, Massachusetts, 1992. Special issue of *Computer Graphics*.
- [7] Barry Fowler and Richard Bartels. Constraint based curve manipulation. *IEEE Computer Graphics and Applications*, 13(5):43–49, September 1993.

- [8] Phil E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.
- [9] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1989. Second edition.
- [10] Hans Hagen and Georges-Pierre Bonneau. Variational design of smooth rational bézier curves. *Computer Aided Geometric Design*, 8(5):53–60, November 1991.
- [11] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, July 1992. Proceedings of SIGGRAPH '92.
- [12] Henry P. Moreton and Carlo H. Séquin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):167–176, July 1992. Proceedings of SIGGRAPH '92.
- [13] Horst Nowacki and Xinmin Lü. Faring composite polynomial curves with constraints. *Computer Aided Geometric Design*, 11(1):1–16, February 1994.
- [14] David F. Rogers and Nils G. Fog. Constrained B-spline curve and surface fitting. *Computer-Aided Design*, 21(10):641–648, December 1989.
- [15] Gilbert Strang. *Linear Algebra and its Applications*. Harcourt, Brace, Jovanovich, San Diego, California, third edition, 1988.
- [16] William Welch and Andrew Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, July 1992. Proceedings of SIGGRAPH '92.
- [17] Andrew Witkin, Kurt Fleischer, and Alan Barr. Energy constraints on parameterized models. *Computer Graphics*, 21(4):225–229, July 1987. Proceedings of SIGGRAPH '87.