# 5.0 Surface Parameterization and Mesh Movement

A key aspect in any design optimization procedure is *how* the design surface and computational mesh are to be represented. This selection will ultimately determine (*i*) the type and number of design variables used, (*ii*) the grid adaptation or regeneration method, and (*iii*) the means through which grid sensitivities are calculated. In the following sections, the techniques used in the present work for (*i*), (*ii*), and (*iii*) will be discussed.

## 5.1 Bezier-Bernstein Curves

The method adopted to represent the design surface should have a physical interpretation, illustrate fairness between the design variables, not create spurious or uncontrollable oscillations between the design variables, and possess geometric flexibility in as few design variables as possible. One approach which has these desired attributes, and which has been used with much success for both structural [160] and aerodynamic [9,19,49,155,161,162] shape optimization problems, is referred to as a Bezier-Bernstein surface parameterization. A brief review of Bezier-Bernstein curves follows; a more detailed discussion may be found in reference 163.

Any point on a Bezier curve segment may be expressed by the following parametric function

$$c(u) = \sum_{i=0}^{N} b_i B_{i,N}(u) \qquad u \in [0,1] \tag{5.1}$$

where $b_i$ represent the $N$+1 vertices referred to as Bezier control points, and the blended functions $B_{i,N}(u)$ are given by the $N^{\text{th}}$-degree Bernstein polynomials

$$B_{i,N}(u) = \frac{N!\ u^i\ (1-u)^{N-i}}{i!\ (N-i)!} \tag{5.2}$$

where $u$ is the normalized computational arclength along the curve. The Bernstein polynomials were chosen because they satisfy the properties that (*i*) the curve must pass through the first and last control points and (*ii*) the tangent to the curve at each point may be controlled or specified if desired.

As can be deduced, the Bezier control points are a logical choice for shape design variables. An example of a curve parameterized with Bezier-Bernstein polynomials is depicted in Fig. 5.1. Once the design variables have been selected and the surface parameterization complete, a means of adapting the computational mesh, $X(\beta_k)$, due to changes in the design variable, $\beta_k$, must be employed. The mesh movement strategy used in the current work will be discussed in section 5.3.

## 5.2 Wing Planform Representation

The wing planform parameterization used in the current work is illustrated in Fig. 5.2. The design variables consist of local chord multipliers and setback distances at the cranks and tip. Throughout the design, the semi-span length, root chord, and distances between cranks are fixed. Similar parameterizations have been used and are discussed in Refs. 162 and 164 for design optimization studies utilizing structured grid approaches. An extremely sophisticated wing parameterization method capable of modeling wing-section (airfoil) definitions, taper distribution, sweep, span and spanwise bending, global angle-of-attack, and twist schedule is developed in Ref. 162 and discussed at length in Ref. 9.

## 5.3 Mesh Movement Strategy

For a structured grid, it is possible to augment the surface parameterization technique with an algebraic grid generation method [9] and, thus, derive explicit relationships between the mesh and the design variables. An explicit, differentiable dependence between the mesh and the design variables has the advantages of being CPU efficient and it allows for the analytical determination of grid sensitivities, $\partial X/\partial \beta_k$ . Unstructured grids, on the other hand, due to an inability to define distinct families of grid lines, may not be adapted by these simple algebraic relations.

An alternative approach, previously developed by Batina [165] and duplicated in Refs. 166 and 167, considers the unstructured mesh as a system of interconnecting springs. This system is constructed by representing each edge of each triangle by a tension spring. The stiffness of this spring is assumed inversely proportional to the length of its edge and may be written as

$$k_{ji} = 1.0 \Big/ \left[ \left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2 \right]^{p/2} \tag{5.3}$$

where $p$ is a parameter used to control the stiffness of the spring. Then, for each mesh point, the external forces due to the connecting boundary springs are summed and resolved into Cartesian components. The resulting set of linear systems may be solved for the displacements of each node using several Jacobi iterations

$$\Delta X_j^{n+1} = \frac{\sum k_{ji} \, \Delta X_i^n}{\sum k_{ji}} \tag{5.4}$$

where $i$ is summed over all edges connected to node $j$. The positions of the interior nodes are then updated using the determined displacements.

This iterative method has the advantage of not requiring an excessive amount of memory, but does require an initial guess for the displacements. In the present work, the initial

guess, at the start of the grid adaptation process for a current design, is the final displacements of the adapted mesh from the previous design. In practice, using this unstructured grid adaptation method, an acceptable mesh is usually achieved in 4 to 6 iterations.

Even though it is not used in the present study, there is yet another means through which the computational mesh about a new design may be obtained. This requires the incorporation of an existing grid generation package into the design optimization procedure [139,164,168]. Then, for each new design, the grid is simply regenerated. In order to remain consistent within the optimization process, however, a strict one-to-one correspondence between the computational meshes at different designs should be maintained (i.e., the size of the computational mesh should not change during the design process). A potential difficulty of this approach, as well as for the above spring analogy method, is the determination of grid sensitivities. The manner in which this problem is circumvented, in the present work, is discussed in the following section.

## 5.4 Grid Sensitivity

Efficient and accurate evaluation of grid sensitivities is an extremely important and vital aspect in any design optimization procedure (which uses discrete sensitivity analysis). The technique used to obtain the grid sensitivities from the unstructured grid adaptation procedure results from the direct application of ADIFOR [136]. Here, the subroutines which parameterize the design surface with the variables $\beta_k$, and the subroutines which perform the unstructured grid adaptation to produce the mesh $X(\beta_k)$, are differentiated using ADIFOR. The result is an additional set of subroutines which, upon compilation and execution, will return the grid sensitivities, $\partial X/\partial \beta_k$.

To verify that these sensitivities were indeed correct for each of the design cases, one of the design variables was perturbed, the grid adapted, and the grid sensitivities calculated via ADIFOR generated subroutines. These sensitivities were then compared with those obtained using a central finite-difference. ADIFOR generated grid sensitivities matched finite-difference to approximately 8 significant digits for both the two- and three-dimensional cases. Qualitative results of these comparisons will be shown in the results.