Approximation of Surfaces and Solution of Partial Differential Equations Using B-Splines

Clarence O. E. Burg*

Mississippi State University, Mississippi State, Mississippi, 39762, U.S.A.

Sunil S. Nandihalli †

Mississippi State University, Mississippi State, Mississippi, 39762, U.S.A.

B-Spline curves and surfaces are widely used by CAD systems to represent physical models and from which grids can be built for computational fluid dynamics simulations. These B-Spline entities can also be used in conjunction with flow solvers either to represent features in the solution or to discretize the governing equations. In this paper, the primary use of B-Splines is to represent the free surface about viscous hulls in conjunction with the generation quality viscous grids. After developing the tools for such a representation, B-Splines are used to develop a finite element framework using the basis functions for the B-Spline surfaces as the weight and interpolating functions. The evaluation of the resulting integral is simplified because the spatial derivatives of the B-Spline surfaces can be calculated exactly. These codes are tested on a set of algebraic cases where exact agreement is possible and then for actual free surfaces generated from a submerged three-dimensional hydrofoil, for the Wigley parabolic hullform and for the prototypical naval destroyer DTMB Model 5415 hullform.

Nomenclature

t	Parameter $\in [0,1]$
$C(t)$	B-Spline Curve
k	Degree
i	Index for Control Points
$n+1$	Number of Control Points
P_i	Control Point
$N_{i,k}(t)$	Basis Function
$P_i^r(t)$	Iterative Control Points at Level r
(u,v)	Parameter for Surface $\in [0,1]\times[0,1]$
$B_(u,v)$	B-Spline Surface
$d_{i,j}$	Control Points in Surface
$N_{i,k}(u), N_{j,l}(v)$	Basis Functions for Surface
x,y,z	Spatial Components
$\tilde{x},\tilde{y},\tilde{z}$	Locations on B-Spline Surface
(u_I, v_I)	Parameters for node I in Unstructured Grid
$\phi_{p,q}$	Finite Element Weight Function
$\hat{U}=(U,V,W)$	Velocity Field

*Assistant Research Professor, Computational Simulation and Design Center, 2 Research Blvd., Starkville, MS, 39759, Member AIAA.

†Graduate Research Assistant, Computational Simulation and Design Center, 2 Research Blvd., Starkville, MS, 39759, Student Member AIAA.

I. Introduction

B-Spline curves and surfaces have some intriguing properties that can be applied to the solution of partial differential equations and the examination and analysis of the features within the solution. One such property is that the smoothness of the B-Spline curve or surface is controlled by the knot vector and the degree of the B-Spline entity. Hence, smoothing the solution of a partial differential equation or the features within a solution occurs naturally when approximating via a B-Spline entity. This characteristic is of particular interest for free surface tracking simulations around ships, where the grid must be moved to match the free surface. If the free surface has high-frequency oscillations or noise, then the grid quality from a flow solver perspective will be quite poor when the grid is moved to match the free surface, as is shown in Figure 1. In practice, the flow solver will often become unstable if the grid at the free surface has this level of roughness. This high frequency noise can be significantly reduced by Laplacian smoothing of the free surface elevations, but it can be completely eliminated by representing the free surface as a smooth B-Spline surface, and the stability of the solution on the grid produced by the grid movement process is improved. Burg, etal[1] has used this smoothing characteristic, as well as the inherent ability to symmetrize a B-Spline surface, to improve the grid quality for their free surface algorithm, especially for flows with transom sterns[2,3].

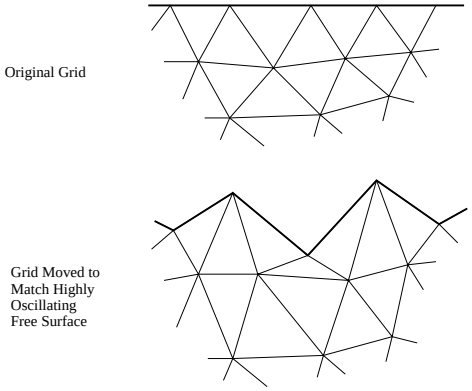

Figure 1. Example of Grid Distortion due to Free Surface.

Another benefit of representing the free surface as a B-Spline surface is the ability to use this B-Spline surface in the grid generation process. NURBS and B-Spline curves and surfaces are native CAD types and can be easily imported into most geometry manipulation packages. For structured grid generators, the volume grid can then be created around the curved initial free surface just as easily as if it were a planar surface. For unstructured grid generators, such as AFLR3[4], any non-viscous surface that intersects the viscous surface must be planar, so that the prismatic boundary layer can terminate cleanly in the free surface which is the surface at the top of Figure 2. The quadrilateral elements near the viscous surfaces on the upper right are the boundary layer elements. The boundary layer elements, which are typically prisms, are generated by starting at the viscous surface using a point spacing consistent with the Reynolds number and grow in thickness outward from the viscous surface. For fully enclosed objects such as a submarine or an airplane, the boundary layer wraps completely around the object, but for surface vessels, the boundary layer intersects the free surface.

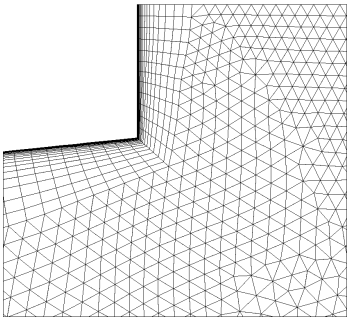

Figure 2. Viscous Grid Terminating into Free Surface.

If the free surface were represented by a general curved surface and not a B-Spline or NURBS surface, the boundary layer could not terminate appropriately. However, if it were represented by a mathematically defined surface such

as a B-Spline surface, then that surface could be transformed into a planar surface for the boundary layer grid generation step, and reverted back to the curved surface, for the remainder of the grid generation and quality improvement steps. The first step is to generate a reasonable approximation to the free surface, as determined from previous simulations, experimental results or expectations, which is shown as the expected free surface in Figure 3. Then, this guess for the free surface is approximated as a B-Spline representation via the algorithm presented herein. The free surface in the region of interest is replaced with this B-Spline representation, which is a mathematically defined surface. Hence, when the volume grid is built, the grid generator uses the B-Spline definition to deform the surface grid in the region of the interest, so that the free surface is now planar, and the viscous surfaces are adjusted via the same definition as is shown in Figure 4. Since the free surface is planar, a good quality prismatic boundary layer grid can be built, that terminates cleanly into the free surface. After the boundary layer grid is built, the grid generator moves the surface and the grid back to its original shape via the inverse B-Spline transformation, the isotropic (tetrahedral) grid is built, and quality improvements are made.

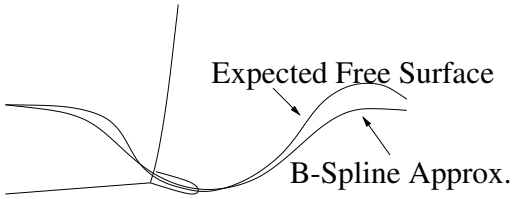

Figure 3. Initial Guess for Free Surface and B-Spline Approximation.

This process has been used in conjunction with a less sophisticated B-Spline approximation algorithm, for grids around a prototypical naval destroyer DTMB Model 5415 hullform. This type of vessel has a transom stern, which is similar to the stern of a row boat rather than of a canoe which tapers to a point. In the region of the transom stern, the free surface may either remain attached to the transom stern, or it may come off the bottom of the hull cleanly. In the former case, a planar free surface can be deformed to match the free surface, as is shown in numerous papers[2,5,6]. However, if the ship is traveling fast enough that the free surface comes cleanly off the bottom of the hull, then the grid must be built using a curved free surface. The algorithm described above and in Figures 2 and 3 was developed to handle this specific case, but it can also be used for other types of problems.

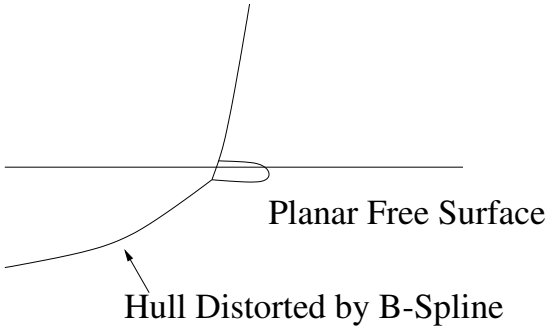

Figure 4. Deformed Surfaces Based on B-Spline Approximation.

Finally, one of the most intriguing properties of B-Spline surfaces is that their spatial derivatives can be calculated exactly. Since they are smooth functions with exact mathematical definition, their derivatives can be calculated up to the degree of the B-Spline. Hence, many of the terms within a partial differential equation can be expressed exactly. Using these B-Spline curves and surfaces within a finite element framework, a novel approach to solving partial differential equations can be derived. If the solution to the PDE is smooth, then the B-Spline representation of the solution should be reasonably accurate. However, in regions with large variations in the solution variables, such as hydraulic shocks, the representation may smear the solution too much. Using the same tools that were developed to approximate a set of points as a B-Spline surface, a finite element solver of the kinematic free surface equation was developed and will be presented herein.

The approach of using B-Spline surfaces within a finite element framework to solve systems of partial differential equations is not a new idea. Gardner, etal[7,8], used cubic B-Splines within a Galerkin finite element method to solve the regularized long wave equation. They were able to demonstrate mathematically that their scheme was unconditionally stable. Ali, etal[9], used the B-Spline finite element method (BSPFEM) to solve the nonlinear Burgers equation. They also showed that a Crank-Nicholson scheme for these equations was unconditionally stable. In a more closely related

work, Verma[10] investigated the use of a quadratic B-Spline finite element method to solve free surface problems for two-dimensional flows, which have a one-dimensional free surface.

In the next section, the relevant aspects of B-Spline curves and surfaces are reviewed. Then the method that was used to form the mapping between the location of the nodes in the unstructured grid and the parametric values in the B-Spline surface is presented. In the following section, the algorithm that was used to approximate a set of data points into a B-Spline surface is described and results are given. In the final section, the algorithm used to solve the kinematic free surface equation is presented, followed by the results.

II. B-SPLINE ESSENTIALS

A B-Spline curve of degree k with a set of $n+1$ control points $P_0, P_1, ...P_n$ and knot vector $t_0, t_1, ..., t_{n+k}$ of nondecreasing sequence of values with $t_i \in [0,1]$ is defined as a weighted average of basis functions $N_{i,k}$ times each control point via the following formula:

$$C(t) = \sum_{i=0}^{n} N_{i,k}(t) P_i \tag{1}$$

where the basis functions $N_{i,k}$ are determined iteratively from

$$N_{i,k}(t) = \frac{(t - t_i)}{(t_{i+k-1} - t_i)} N_{i,k-1}(t) + \frac{(t_{i+k} - t)}{(t_{i+k} - t_{i+1})} N_{i+1,k-1}(t) \tag{2}$$

with termination criterion that

$$N_{i,1}(t) = \begin{cases} 1 & \text{for } t_i \leq t < t_{i+1}, \\ 0 & \text{otherwise .} \end{cases} \tag{3}$$

The de Boor algorithm is a recursive means for evaluating the B-Spline curve at a particular location t in the knot vector, where $t \in [t_i, t_{i+1})$ and $P_i^0(t) = P_i$. Then, the value of the curve $C(t)$ is defined by recursively evaluating the following formula:

$$P_i^r(t) = \frac{t_{i+k+r} - t}{t_{i+k+r} - t_{i-1}} P_{i-1}^{r-1} + \frac{t - t_{i-1}}{t_{i+k+r} - t_{i-1}} P_i^{r-1} \tag{4}$$

and setting $C(t) = P_i^k(t)$. This algorithm is described in full detail in Farin[11]. This recursive algorithm is a more efficient means for evaluating the function than evaluating each of the basis functions $N_{i,k}$, for each different location in the knot vector. However, if the B-Spline curve is to be repeatedly evaluated using the same location in the knot vector, then it is more efficient to calculate and store the basis functions, and reuse these basis functions for the evaluation of the different curves and surfaces, directly via equation (1). To evaluate these basis functions, the de Boor algorithm can be used by setting all of the control points to 0, except for the control point for the basis function being evaluated, which is set to 1.

One of the properties of Bezier, B-Spline and NURBS curves and surfaces is that the derivative of the curve with represent to the parameter t can be calculated exactly, as

$$\frac{dC(t)}{dt} = \sum_{i=1}^{n} N_{i,k-1}(t) P_i' \tag{5}$$

using the derivative control points P_i^1 which are defined as

$$P_i' = \frac{P_i - P_{i-1}}{t_{i+k-1} - t_{i-1}} \tag{6}$$

Equivalently, the derivative of the B-Spline curve can be determined by modifying the basis functions, or

$$\frac{dC(t)}{dt} = \sum_{i=1}^{n} \frac{\partial N_{i,k}(t)}{\partial t} P_i \tag{7}$$

where

$$\frac{\partial N_{i,k}(t)}{\partial t} = \begin{cases} 0 & \text{if } k = 1, \\ \frac{N_{i,k-1}(t)+(t-t_i)\frac{\partial N_{i,k-1}(t)}{\partial t}}{t_{i+k-1}-t_i} - \frac{N_{i+1,k-1}(t)+(t_{i+k}-t)\frac{\partial N_{i+1,k-1}(t)}{\partial t}}{t_{i+k}-t_{i+1}} & \text{otherwise .} \end{cases} \tag{8}$$

Evaluating the derivative using Equation (5) is more efficient if the derivative will be evaluated at various locations in the curve using the same control points, but using Equation (7) is more efficient if the locations for the evaluation do not change but the value of the control points do change, as is the case for the algorithms developed herein. By evaluating the B-Spline curves and surfaces in the more efficient manner, the computational cost for several of these algorithms is reduced by an order of magnitude.

Another type of derivative that will be used herein is the derivative of the B-Spline curve with respect to the control points, or $\frac{\partial C(t)}{\partial P_i}$. These derivatives result in the basis functions as is shown below. Again, since the parametric locations are fixed, these derivatives need to be calculated only once and then are stored.

$$\frac{\partial C(t)}{\partial P_i} = N_{i,k}(t) \tag{9}$$

Now that the basics for a B-Spline curve have been presented, the formulation of B-Spline surfaces are presented. These surfaces are evaluated in the same manner as the curves are evaluated, as well as their derivatives. Given a set of $(m+1) \times (n+1)$ control points $d_{i,j}$, two different degrees k and l, and two different knot vector $u_0, u_1, ..., u_{m+k}$ of nondecreasing sequence of values with $u_i \in [0,1]$ and $v_0, v_1, ..., v_{n+l}$ of nondecreasing sequence of values with $v_i \in [0,1]$, a B-Spline surface can be built using two parameters u and v, via a product of B-Spline curves, or

$$B(u,v) = \sum_{i=0}^{m} \left[N_{i,k}(u) \left(\sum_{j=0}^{n} N_{j,l}(v) d_{i,j} \right) \right] \tag{10}$$

where the basis functions $N_{i,k}(u)$ and $N_{j,l}(v)$ are defined as above.

Within the following derivations, two different derivatives will be needed - the derivative of the B-Spline surface with respect to the parameters u and v and the derivative of the surface with respect to the control points $d_{i,j}$. Clearly, the derivative of the B-Spline surface with respect to $d_{i,j}$ is only dependent on the basis functions and hence can be determined once and then stored, or

$$\frac{\partial B(u,v)}{\partial d_{i,j}} = N_{i,k}(u) N_{j,l}(v) \tag{11}$$

Following the derivation given above, the derivative with respect to the parameter u is

$$\frac{\partial B(u,v)}{\partial u} = \sum_{i=0}^{m} \left[\frac{\partial N_{i,k}(u)}{\partial u} \left(\sum_{j=0}^{n} N_{j,l}(v) d_{i,j} \right) \right] \tag{12}$$

where the derivative of the basis function mimics Equation (8). The derivative of the B-Spline surface with respect to the parameter v is defined in an analogous fashion.

Finally, to represent a surface in 3D space, a point on this surface given by the parameters (u,v) is given by the triplet $(x(u,v), y(u,v), z(u,v))$ where each coordinate is given by a set of control points. For the examples presented herein, the x and z-coordinates are determined by an O-grid or a rectilinear structured grids, and the y-coordinates are determined by the elevation of the free surface or by the velocity field associated with the free surface.

In order to calculate the derivatives $\frac{\partial y}{\partial x}$ and $\frac{\partial y}{\partial z}$, which are needed for the kinematic free surface solver, the chain rule is applied to each term to get the following matrix relationship:

$$\begin{bmatrix} \frac{\partial y}{\partial x} \\ \frac{\partial y}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} \end{bmatrix} \begin{bmatrix} \frac{\partial y}{\partial u} \\ \frac{\partial y}{\partial v} \end{bmatrix} \tag{13}$$

The terms $\frac{\partial y}{\partial u}$ and $\frac{\partial y}{\partial v}$ can be calculated from expressions given above, such as equation (12). But, the terms in the matrix are not defined above. By using the chain rule, it can be easily shown that this matrix can be defined as is shown below, completing the definition of these derivatives.

$$\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial z}{\partial v} \end{bmatrix}^{-1} \tag{14}$$

III. TAGGING THE NODES

Given a two-dimensional unstructured grid or set of data points to be fit with an approximating surface and a two-dimensional structured B-Spline surface, where the x and z control points are fixed, a mapping between the (x, z) coordinates of each point in the unstructured grid and its parametric values within the B-Spline surface needs to be established. In this derivation, the y-coordinates for the B-Spline surface are the unknowns. Gardner, etal [7,8,9], developed a method where the x, y and z-control points were the unknowns, allowing for greater flexibility, but for the applications presented herein (i.e., free surface flows around surface ships), this level of flexibility and computational complexity is not required.

Given a B-Spline surface with $m+1 \times n+1$ control points in the XZ plane, the values of x and z coordinates at parameters $0 \leq u, v \leq 1$ can be calculated via

$$\begin{aligned} \tilde{x}(u,v) &= \sum_{i=0}^{m} N_{i,k}(u) \left(\sum_{j=0}^{n} N_{j,l}(v) X_{i,j} \right) \\ \tilde{z}(u,v) &= \sum_{i=0}^{m} N_{i,k}(u) \left(\sum_{j=0}^{n} N_{j,l}(v) Z_{i,j} \right) \end{aligned} \tag{15}$$

where $X_{i,j}$ and $Z_{i,j}$ are the x and z control points. For a node in the unstructured grid or a point in the set of data points $P_t = (x_t, z_t)$, the parameter values (u_t, v_t) need to calculated such that $x_t = \tilde{x}(u_t, v_t)$ and $z_t = \tilde{z}(u_t, v_t)$. Given an initial guess (u_0, v_0), define the difference between the coordinates defined by this guess and the target coordinates as

$$(dx, dz)^T = (\tilde{x}(u_0, v_0) - x_t, \tilde{z}(u_0, v_0) - z_t)^T \tag{16}$$

Since the goal is to drive (dx, dz) to $(0, 0)$, Newton's method is an appropriate algorithm, which can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix}_{new} = \begin{bmatrix} u \\ v \end{bmatrix}_{old} - \begin{bmatrix} x_u & x_v \\ z_v & z_v \end{bmatrix}^T \begin{bmatrix} dx \\ dz \end{bmatrix} \tag{17}$$

where x_u, x_v, z_u, z_v represent the derivatives of $(\tilde{x}(u,v), \tilde{z}(u,v))$ with respect to the parametric coordinates (u, v). These derivatives can be evaluated via equation (5). This algorithm is continued until the magnitude of the vector $(dx, dz)^T$ is below some tolerance.

Appropriate restrictions are needed to ensure that the parameter locations $(u, v)^T_{new}$ lie within the domain $[0, 1] \times [0, 1]$. Additionally, if the grid is an O-grid or C-grid, then other modifications are needed along the re-entrant boundaries, if the new parameter location crosses over the boundaries.

Finally, to make this algorithm robust and efficient, a good initial guess is needed. An average of the parametric values for the neighboring nodes, whose values are already known, serves an excellent choice. To take advantage of this observation, the order in which the parametric values of the nodes is calculated should be slightly modified.

- Nodes with known parametric values.
- ○ Nodes in list.
- □ Nodes added to list.

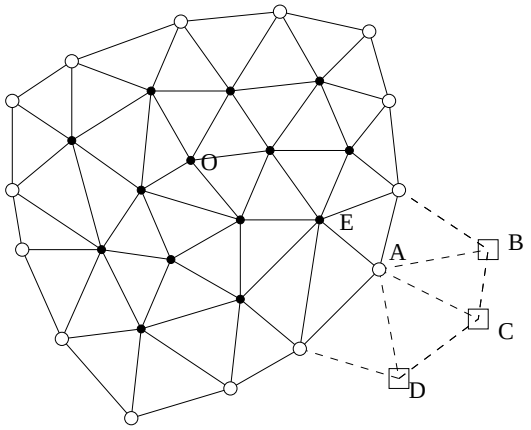

Figure 5. Efficient Tagging of Nodes.

After calculating the value of the first node in the unstructured grid, using an initial guess of (0.5,0.5), the nodes connected to that node are placed in a list. The first node in this list is chosen as the next node whose mappings is to be determined. Taking the average of the parametric values for its neighboring nodes that have already been determined as the initial guess, the parametric values for this node is determined via the above algorithm, then the neighboring nodes for this node are added to the list, unless they are already in the list. The next node in the list is chosen, and the search is continued. This algorithm is depicted in Figure 5. This modified ordering greatly improves the efficiency of the process of tagging the nodes, due to the reduction in the number of iterations needed for each node.

IV. B-SPLINE APPROXIMATER

In this section, the method used to approximate a surface defined via the elevations of the y-coordinates in a three-dimensional unstructured surface mesh to a B-Spline approximating surface is described. This algorithm is described and then tested on an algebraic test case, where the target surface is exactly attainable by the B-Spline representation, and then tested on the free surface generated about three different hullforms. Both rectilinear and O-grids are used for the underlying B-Spline surface, where appropriate.

A. B-Spline Approximation Algorithm

For each node in the unstructured grid, given by coordinates (x_I, y_I, z_I), the parametric value (u_I, v_I) within the B-Spline surface has been determined, as described above, so that $\tilde{x}(u_I, v_I) = x_I$ and $\tilde{z}(u_I, v_I) = z_I$. Thus, the goal is determine the control points in the y-coordinate direction that minimize the error between the B-Spline surface in the y-direction and the y-values at the nodes in the unstructured grid, as defined by the function $F(Y_{i,j})$ where $Y_{i,j}$ are the y-value of the control points,

$$F(Y_{i,j}) = \sum_{I=0}^{N_{nodes}} \left(\tilde{y}(u_I, v_I) - y_I\right)^2 A_i \tag{18}$$

where y_I is the elevation at node I and $\tilde{y}(u_I, v_I)$ is the value of B-Spline surface for the parametric value (u_i, v_I) as is defined as

$$\tilde{y}(u, v) = \sum_{i=0}^{m} N_{i,k}(u) \left(\sum_{j=0}^{n} N_{j,l}(v) Y_{i,j} \right) \tag{19}$$

Thus, to minimize this function, the derivative with respect to each control point must be zero, which is

$$\frac{\partial F(Y_{i,j})}{\partial Y_{p,q}} = \sum_{I=0}^{N_{nodes}} 2 \frac{\partial \tilde{y}(u_I, v_I)}{\partial Y_{p,q}} \left(\tilde{y}(u_I, v_I) - y_I\right) A_i \tag{20}$$

The term $\frac{\partial \tilde{y}(u_I, v_I)}{\partial Y_{p,q}}$ reduces to the basis function $N_p, k(u_I) N_q, l(v_I)$ so that equation (20) becomes

$$\sum_{I=0}^{N_{nodes}} N_{p,k}(u_I) N_{q,l}(v_I) \left(\tilde{y}(u_I, v_I) - y_I\right) A_i = 0 \tag{21}$$

If the goal is to minimize the function $\Delta\tilde{y}(u, v)$ which is a continuous function representing the difference between the B-Spline surface $\tilde{y}(u, v)$ and the target free surface, the Galerkin finite element formulation is

$$\int_{\Omega} \phi_{p,q}(u, v) \Delta\tilde{y}(u, v) d\Omega = 0 \tag{22}$$

Evaluating this integral at each nodal location (u_I, v_I) within the unstructured grid, this integral can be discretized as

$$\sum_{I=0}^{N_{nodes}} \phi_{p,q}(u_I, v_I) \Delta y_I A_I \tag{23}$$

where $\Delta y_I = \tilde{y}(u_I, v_I) - y_I$ and A_I is the area of the control volume associated with each node I. By choosing $\phi_{p,q}$ to be the B-Spline basis functions $N_p, k(u) N_q, l(v)$, these formula are equivalent.

These equations are solved in iterative form, using $DQ - GMRES$ [9].

B. Void Space

For surface vessels that intersect the water surface, the interior of the ship (i.e., the void space) has no influence on the free surface, and the free surface has no influence on the control points of the B-Spline surface which lie within the void space. Thus, another means must be developed for determining these control points, in order for the surface to vary smoothly throughout the void space. Distance weighted averaging based on the X and Z control points of the B-Spline mesh was used to blend the Y control points values at the edge of the void space.

V. Validation of B-Spline Approximation Algorithm

A. Algebraic Test Case

In order to test the B-Spline approximation algorithm, an algebraic test case was contrived that would admit an exact B-Spline solution. An algebraic function was chosen to provide the target B-Spline control points. The Y control points were evaluated using this algebraic function. Using these target control points, the elevations in an unstructured square grid were calculated via the B-Spline definition and stored as the target elevations Y_I. The target elevations are shown in Figure 6.

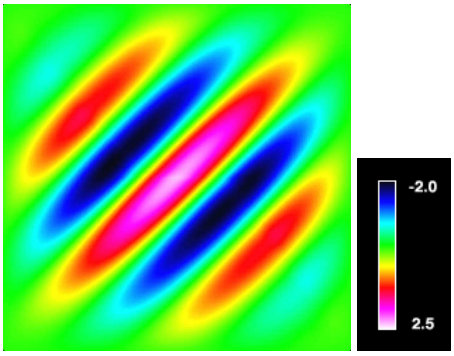

Figure 6. Target Elevations for Algebraic Case.

Both an O-Grid and a rectilinear B-Spline grid were used to reconstruct the surface. The unstructured grid consisted of 56,535 nodes, while the O-grid was 20×50 and the rectilinear grid was 20×20. For the O-Grid case, the approximation algorithm was able to reconstruct the Y control points of the B-Spline surface to three orders of magnitude as is shown in Figure 7; and for the rectilinear case, the approximation algorithm reconstructed the surface to machine precision as presented in Figure 8. The reason for the degradation in the reconstruction for the O-grid deals with the boundary condition across the singularity line. A typical O-grid used for the B-Spline approximation algorithm is shown in Figure 9. Often O-grids are used to fit a mesh around an airfoil or some other object in the center of the grid; for this application, no such object is present, which means that the center of the grid is a degenerate curve, which must be treated in a special fashion. The reason that an O-grid is used rather than a rectilinear grids is that the points for the O-grid can be clustered near the hull of the ship.

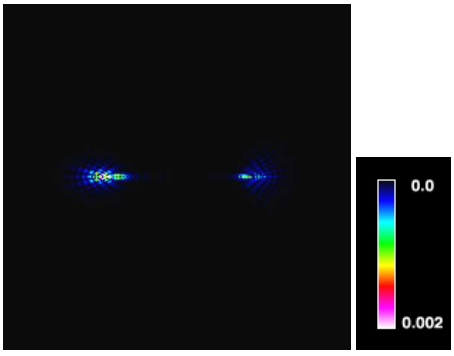

Figure 7. Error for Algebraic Case Using O-Grid.

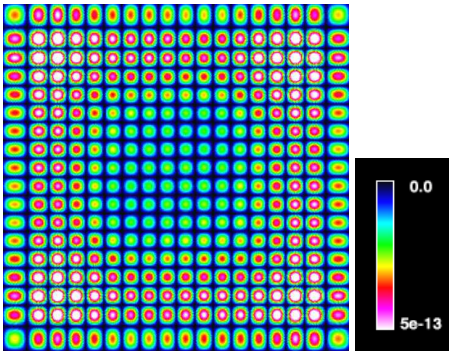

Figure 8. Error for Algebraic Case Using Rectilinear Grid.

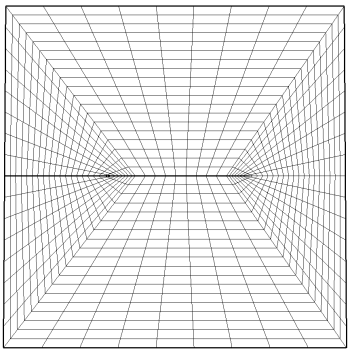
Figure 9. Typical O-Grid.

B. Physically Realistic Cases

The results for the physically realistic cases show that the B-Spline approximation algorithm can adequately reconstruct the general features of the free surface. However, due to the inherent smoothing that occurs for B-Spline curves, the accuracy of the reconstruction will be less in regions of large variation in the free surface. By using an O-grid, which will allow for more control points near the hull where the free surface is varying the most, the accuracy of the representation is improved. The size of the B-Spline surfaces and the number of nodes in the unstructured surface grids are shown in Table 1. The ratio of the number of nodes in the unstructured grid to the number of control points is approximately 10 to 1, showing that the number of unknowns is significantly reduced in this procedure.

Case	B-Spline Control Pts	Grid Type	Nodes in Surface
3D Hydrofoil	30×81	O-grid	27,543
3D Hydrofoil	49×76	Rectilinear	27,543
Wigley Hull	40×101	O-grid	68,874
DTMB 5415	69×109	O-grid	83,292

Table 1. Grid Sizes for Test Cases.

The first physically realistic case deals with a proprietary three-dimensional hydrofoil, for which we completed a viscous free surface simulation several years ago. Since the hydrofoil is fully-submerged and since it was a three-dimensional flow field, this case was a good next step for the algorithm. The target elevations are provided in Figure 10, with the flow from left to right and the submerged hydrofoil just below the white region of the free surface. At the boundary of the domain, the elevations are not zero, although the B-Spline approximation algorithm fixes the boundaries to zero. This difference is apparent in Figures 11 and 12 which show the error between the B-Spline surface and the target surface for an O-grid and a rectilinear grid. The errors exist in four different locations: above the hydrofoil where the free surface variation is the greatest, along the centerline, at the side boundaries shown on the top and the bottom of the images, and at the outflow boundary along the right of the image. The first type of error is due to the inherent smoothing when using B-Spline curves and surfaces and is expected. The second type, which deals with the error along the centerline or symmetry plane, is much more noticeable for the O-grid which treats the

centerline as a flattened circle. This error is due to the enforcement of continuity across this centerline, and for typical monohull surface vessels, this flattened circle will lie within the void space of the monohull. The third error at the side boundaries is caused by the fact that the B-Spline surface expects the free surface to be zero along these boundaries when they are not.

The fourth type of error, which is at the outflow boundary, is due to both the non-zero free surface at the outflow plane and due to the coarseness of the grid relative to the B-Spline surface mesh. From our experience with approximating free surface elevations on unstructured grids to B-Splines, if the unstructured grid is too coarse relative to the B-Spline mesh, then the B-Spline surface is oscillatory in that region. Hence, if there are too few nodes within the domain of influence of a control point, then the B-Spline approximation algorithm generates erroneous oscillations due to the numerics of the algorithm. This behavior was especially seen in the rectilinear grids at the outer boundaries where the unstructured grid is less refined.

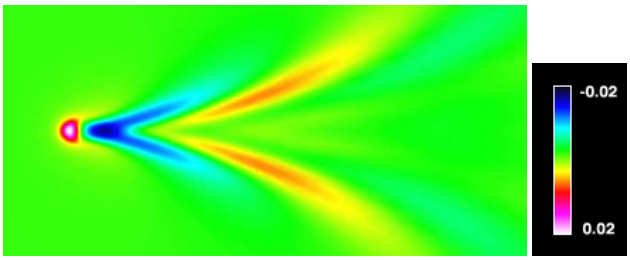

Figure 10. Target Elevations for 3D Hydrofoil.

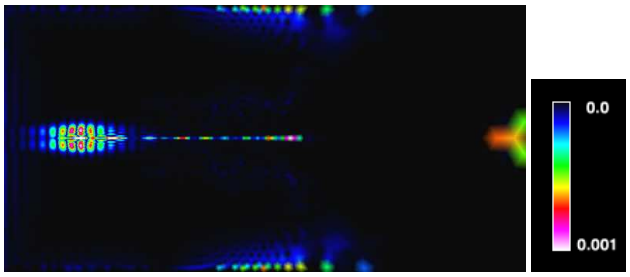

Figure 11. Error for 3D Hydrofoil Using Rectilinear Grid.

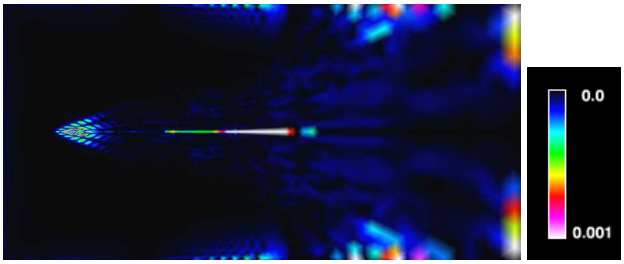

Figure 12. Error for 3D Hydrofoil Using O-Grid.

The second realistic case is the free surface around a prototypical naval destroyer class hullform called the DTMB Model 5415 hull. This ship has a bulbous bow and the transom stern. The transom stern is the region of particular interest because at slow speeds the transom stern is fully-wetted but at high speeds the transom stern is dry. If it is fully-wetted, then the grid at the free surface can be plane that is gradually moved to match the free surface. However, if the transom stern is dry or only partially wetted, then the grid in the stern region must be built to match the expected free surface, which is the motivation for this research.

The free surface elevations about the DTMB Model 5415 at Froude number of 0.28 is shown in Figure 13. For this case, the free surface disturbances reach the top and bottom boundaries as well as the outflow boundary to the right of the image. The computed elevations are shown in Figure 14 and match the target free surface excellently. There are some differences at the bow of the ship and some minor differences in the stern region, but overall the agreement is superb. The errors are shown in Figure 15 and are consistent with the explanations for the Wigley hullform and for the submerged hydrofoil. The scale for Figure 15 is significantly different from the scale for Figures 13 and 14, so that the error between the target and the computed elevations can be seen more clearly.

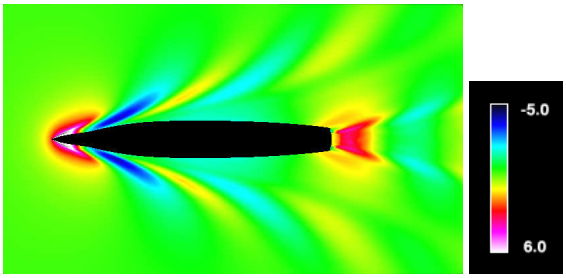

Figure 13. Target Elevations for DTMB 5415.

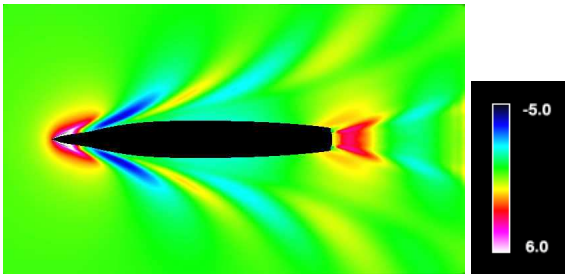

Figure 14. Computed Elevations for DTMB 5415.

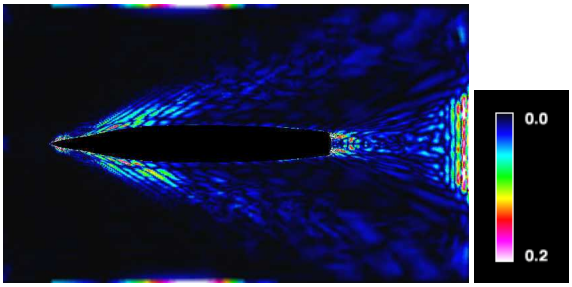

Figure 15. Error for DTMB 5415.

VI. B-Spline Free Surface Solver

The B-Spline free surface solver moves the control points for the B-Spline surface in conjunction with the partial differential equation governing the solution of the free surface (i.e., the kinematic condition), or

$$\frac{\partial y}{\partial t} + U\frac{\partial y}{\partial x} - V + W\frac{\partial y}{\partial z} = 0 \tag{24}$$

Hence, the inputs to the solver are the old free surface control points and the velocity $\hat{U} = (U, V, W)$ at each node in the unstructured grid, and the solver outputs new free surface control points that optimize the solution to this equation via a finite element discretization. In particular, the weight functions and the basis functions for the finite element discretization are the same as the B-Spline basis functions, $N_{i,k}(u)N_{j,l}(v)$. The equation associated with the control point $Y_{p,q}$ is obtaining by multiplying the governing equation by the basis function $N_{p,k}(u)N_{q,l}(v)$ or

$$\int_\Omega N_{p,k}N_{q,l}\left(\frac{\partial \tilde{y}}{\partial t} + U\frac{\partial \tilde{y}^{N+1}}{\partial x} - V + W\frac{\partial \tilde{y}^{N+1}}{\partial z}\right) d\Omega = 0 \tag{25}$$

where N is the time level and Ω is the domain for the B-Spline surface. By making the substitution, $\Delta\tilde{y} = \tilde{y}^{N+1} - \tilde{y}^N$ and using the assumption that $\frac{\partial \tilde{y}}{\partial t} = \frac{\Delta \tilde{y}}{\Delta t}$, then this equation becomes

$$\int_\Omega N_{p,k}N_{q,l}\left(\frac{\Delta\tilde{y}}{\Delta t} + U\frac{\partial \Delta\tilde{y}^{N+1}}{\partial x} - V + W\frac{\partial \Delta\tilde{y}^{N+1}}{\partial z}\right) d\Omega = -\int_\Omega N_{p,k}N_{q,l}\left(\frac{\Delta\tilde{y}}{\Delta t} + U\frac{\partial \tilde{y}^{N}}{\partial x} - V + W\frac{\partial \tilde{y}^{N}}{\partial z}\right) d\Omega \tag{26}$$

where the unknowns are $\Delta\tilde{y}$ and are defined as

$$\Delta\tilde{y}(u,v) = \sum_{i=0}^{m} N_{i,k}(u) \left(\sum_{j=0}^{n} N_{j,l}(v) \left[Y_{i,j}^{N+1} - Yi, j^{N} \right] \right) \tag{27}$$

Equation (26) was discretized by evaluating each integral via a quadrature that uses the information at the nodes of the unstructured grid, similar to the discretization used for the B-Spline approximater. It should be noted that the spatial derivatives on the right-hand-side of the equation can be evaluated exactly by using the derivatives of the B-Spline surface and that the weights associated with the unknowns can be calculated exactly, since they are just combinations of the B-Spline basis functions, as discussed in a previous section. These equations were similarly solved in iterative form, using DQ-GMRES.

A. Physically Realistic Cases

The two physically realistic cases analyzed using the B-Spline free surface solver were the submerged hydrofoil and the Wigley hull. The error for the submerged hydrofoil is shown in Figure 17. The error is symmetric since the velocities generated by the RANS flow solver were symmetric about the symmetric hydrofoil. The error at the outer boundaries is associated with the under-resolution of the unstructured grid as well as the incompatibility in the boundary conditions at the outer boundary. The B-Spline solver forces the free surface to zero at these boundaries, but the velocity field is not consistent with this assumption.

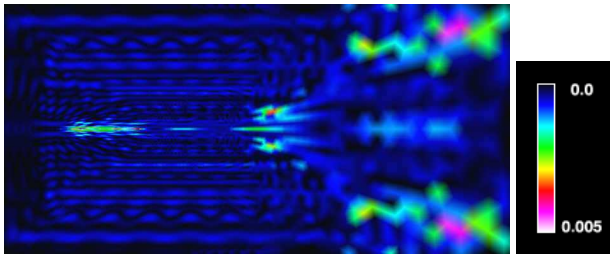

Figure 17. Error for 3D Hydrofoil.

The second physically realistic case deals with the free surface around the Wigley hullform. The Wigley hull is a parabolic surface for which extensive experimental and numerical data have been generated and as such is a typical validation case for free surface flow simulations. Because the Wigley hull crosses the waterline, there is a void space in the middle of the geometry where the free surface does not exist, and the B-Spline control points for the void space must be determined using the alternate algorithm described above.

The free surface for the flow at Froude number 0.289 is shown in Figure 18, with the flow from left to right. The free surface at the left, top and bottom boundaries of the image are close to zero, since the free surface disturbances did not reach these boundaries. The difference between the B-Spline approximation and the target free surface is shown in Figure 19. Since the free surface is close to zero along top and bottom boundaries, there is no appreciable error along these boundaries. There is error near the Wigley hull, especially near the bow and stern of the hull, due to the smoothing effects of the B-Spline representation.

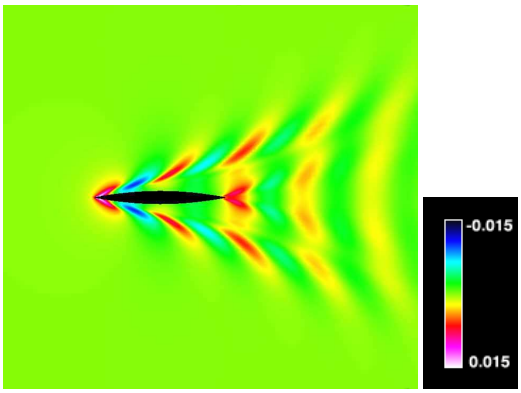

Figure 18. Target Elevations for Wigley Hull.

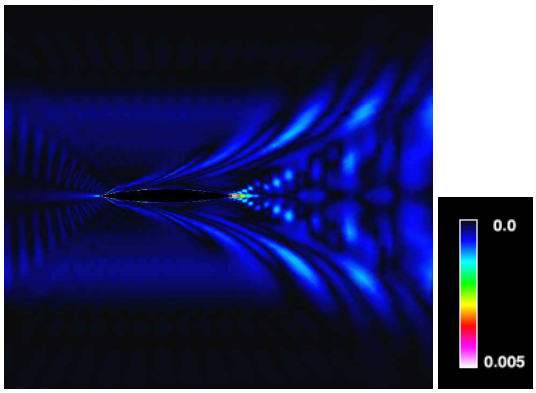

Figure 19. Error for Wigley Hull.

VII. Conclusion

Algorithms for using B-Spline surfaces to represent a free surface for physically realistic geometries and to solve for the free surface for these geometries have been presented. These algorithms can be viewed within the context of the Galerkin finite element method, and for algebraic cases, where the exact solution is realizable, the algorithms were able to converge to the target solution. For the simulations where the free surface and the velocity field were generated by solving for the flow around a ship's hull, the algorithms were able to capture the pertinent features well, smoothing the results, in an advantageous fashion. Unfortunately, the free surface and the velocity field at the edges of the B-Spline surface were not consistent with an undisturbed free surface, so that the boundary conditions for the B-Spline approximater and the B-Spline free surface solver were erroneous. Nonetheless, near the hull, the B-Spline representations were quite accurate.

Future work includes incorporating these algorithms within the grid generator AFLR3 and within the unstructured flow solver U^2NCLE, as well as improving the boundary conditions, as stated above. This type of algorithm has been successfully used to generate viscous unstructured grids for the transom stern of the DTMB Model 5415 geometry as well as for the DDG-51 hullform, when the stern flap is present. Given the current viscous unstructured grid generation algorithms, this grid generation approach using a B-Spline representation for the free surface provides a useful option for building high quality viscous grids. Continued testing and application of these algorithms for realistic geometries is expected.

Acknowledgments

The authors would like to acknowledge and thank Patrick Purtell of the Office of Naval Research and Robert Beck of the University of Michigan for their oversight and funding of this project.

References

[1]Burg, C. O. E., Sreenivas, K., Hyams, D. G., and Mitchell, B., *Unstructured Nonlinear Free Surface Flow Simulations: Validation and Verification*, AIAA Paper 2002-2977, 32nd AIAA Fluid Dynamics Conference and Exhibit, St. Louis, MO, June, 2002.

[2]Burg, C. O. E., Sreenivas, K., Hyams, D. G., and Mitchell, B., *Unstructured Nonlinear Free Surface Simulations for the Fully-Appended DTMB Model 5415 Series Hull Including Rotating Propulsors*, Proceedings of the 24th Symposium on Naval Hydrodynamics, Fukuoka, Japan, July, 2002.

[3]Burg, C. O. E., Marcum, D. L., *Moving Towards High-Fidelity RANS Calculations of Maneuvering Surface Vessels Using Unstructured Grids*, Proceedings of the 8th International Conference on Numerical Ship Hydrodynamics, Busan, Korea, Sept., 2003.

[4]Marcum, D. L., *Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection*, The Handbook of Grid Generation, edited by J. F. Thompson, B. Soni and N. Weatherill, CRC Press, 1998, Section 18.

[5]Beddhu, M., Jiang, J. Y., Whitfield, D. L., and Taylor, L. K., *Computation of the Wetted Transom Stern Flow over Model 5415*, Proceedings of the 7th International Conference on Numerical Ship Hydrodynamics, Nantes, France, 1999.

[6]Wilson, R., Stern, F., *Unsteady RANS Simulation of a Surface Combatant with Roll Motion*, Proceedings of the 24th Symposium on Naval Hydrodynamics, Fukuoka, Japan, July, 2002.

[7]Gardner, L.R.T., Gardner, G. A., *Solitary Waves of the Regularised Long-Wave Equation*, J. Computational Physics, 1990.

[8]Gardner, L. Gardner, G. A., and Dag, I., *A B-Spline Finite Element Method for the Regularised Long-Wave Equation*, Communications in Num. Methods in Eng., 1995.

[9]Ali, A., Gardner, G. and Gardner. L., *A Collocation Solution for Burgers Equation using Cubic B-Spline Finite Elements*, Computer Meth. in

Applied Mech. and Eng., 1992.

[10] Verma, R., *Geometry Modeling Techniques for Free Surface Flows*, Master's Thesis, Mississippi State University, 2001.

[11] Farin, G., Curves and Surfaces for Computer Aided Geometric Design, Academic Press Limited, London, 1990.

[12] Saad, Y., and Wu, K., *DQGMRES: A Direct Quasi-Minimal Residual Algorithm Based on Incomplete Orthogonalization*, Technical Report, Univ. of Minnesota, Computer Science Department, 1995.