INEEL/CON-99-00180
PREPRINT

A B-Spline-Based Colocation Method to Approximate the Solutions to the Equations of Fluid Dynamics

R. W. Johnson
M. D. Landon

July 18, 1999 – July 23, 1999

3rd ASME/JSME Joint Fluids Engineering Conference

Proceedings of the:
3rd ASME/JSME Joint Fluids Engineering Conference
July 18–23, 1999, San Francisco, California

A B-SPLINE BASED COLLOCATION METHOD TO APPROXIMATE THE SOLUTIONS TO THE EQUATIONS OF FLUID DYNAMICS

Richard W. Johnson and Mark D. Landon
Idaho National Engineering and Environmental Laboratory
Idaho Falls, ID 83415-3805

Keywords: B-spline, numerical method, collocation, CFD

ABSTRACT

The potential of a B-spline collocation method for numerically solving the equations of fluid dynamics is discussed. It is known that B-splines can resolve complex curves with drastically fewer data than can their standard shape function counterparts. This feature promises to allow much faster numerical simulations of fluid flow than standard finite volume/finite element methods without sacrificing accuracy. An example channel flow problem is solved using the method.

INTRODUCTION

The amount of time required to obtain a numerical solution is one of the most important criteria by which the usefulness of computational fluid dynamics (CFD) is judged by its users. These include basic researchers who use direct numerical simulation (DNS) to compute the details of turbulence, combustion or other physical phenomena; commercial code developers; design engineers; and particularly designers who employ optimization with analysis to optimize designs. An important consideration for design and optimization engineers is the overall design time required wherein a number of CFD simulations are necessary to find the optimal design. At the other end of the spectrum are those who are engaged in DNS of fluid flow and want to push the envelope (increase the Reynolds number) for which DNS calculations are possible. Of course, the trade-off to speed is accuracy, and CFD users require that numerical CFD solutions exhibit acceptable accuracy.

Solution speed is affected by the speed of the computer used for the simulation, especially the level to which the code is optimized for the specific hardware, including the level to which the solution process has been parallelized when parallel cpus are available. The speed is also a function of the efficiency and stability of the numerical solution algorithm and the efficiency of the numerical method used.

The present paper examines the potential for a new approach to approximating solutions to the equations of fluid dynamics for use in CFD. Although the method can technically be classified as a finite element collocation method, it was arrived at through a totally different thought process and is more akin to the fields of Computer Aided Design (CAD) and Computer-Aided Geometric Design (CAGD). In the fields of CAD/CAGD, curves and surfaces are commonly represented through the use of Bezier, B-spline, and NURBS (nonuniform rational B-spline) curves and surfaces. The curves and surfaces that are defined by these versatile and flexible advanced geometric functions are typically used to represent the geometry of solid objects. In fact, Bezier curves were originally created to mathematically define arbitrary curves and surfaces that automobile designers created for automobile shapes so that numerically controlled machines could be used in the manufacturing process.

The obvious connection between advanced geometric functions that can represent arbitrary geometric shapes and CFD is that the solution to the equations of fluid dynamics can be thought of as curves in 1-D, surfaces in 2-D and volumes in 3-D. We seek B-spline curves, surfaces or volumes that represent the solutions to the fluid equations. Because n-degree B-spline curves and surfaces are defined by very few data points and are C^{n-1} continuous in terms of their parametric derivatives, we hope that we can reduce the amount of computational effort required to find numerical solutions. Furthermore, we can achieve any level of accuracy we choose by choosing the degree of the curve and the number of splines that constitute a curve or surface. Finally, we need not define a mesh for the solution process in the usual sense. In fact, we will not integrate the fluid equations; we can simply evaluate the B-spline functions to obtain all of the terms that appear in the equations. This last feature of the method is makes it a collocation method.

A finite element collocation method is characterized by employing the Dirac delta function as the test function and then integrating in the usual FEM manner, see Carey and Oden (1983), and Lapidus and Pinder (1982). Using the Dirac delta function for the test function has the effect of removing the integral, leaving only the original differential equation. Because there is no integration to perform on some finite element, the differential equation must be evaluated on some network of points; these are called collocation points. Of course, the idea of simply evaluating curve or surface values and their derivatives for some approximating curve or surface need not have originated from within the finite element method.

As mentioned above, B-spline surfaces are used within the CAD community to describe arbitrarily shaped surfaces. A technique that has been used in this field is to obtain a partial differential equation whose solution yields some desired surface. That is, they search for a PDE to represent a desired surface for design purposes; this is called the PDE method. Brown et al (1998) and Brown et al (1990) show how this PDE method can be implemented by using B-splines as basis functions to solve the PDE using a Galerkin finite element method. Bloor and Wilson (1990) show how to apply the PDE method using B-splines with a collocation method. The latter is basically the same idea presented herein, except that our goal is to find a solution to the PDE, rather than to find a PDE to yield a desired surface.

The value of B-splines has also received some attention in the numerical analysis community. Shariff and Moser (1998) and Kravchenko, Moin and Moser (1996) have used B-splines as basis functions in a Galerkin method to resolve near wall eddies in LES/DNS. They embed special meshes near the wall where the B-splines are used. They also discuss the attractive features of using B-splines, including the arbitrary order of accuracy and high resolution attainable, drastic reduction of numbers of grid points and automatic C^{n-1} continuity for n-degree B-splines. Finally, B-splines are have also been used with the Boundary Element Method in a transient 3-D elastodynamic analysis, Rizos and Karabalis (1998).

BEZIER CURVES AND B-SPLINE CURVES

In the late 1950's and early 1960's Pierre Bezier of Renault and Paul de Casteljau of Citroen, independently created what are now termed Bezier curves, Farin (1997). We describe some of the features of Bezier curves and B-splines for the convenience of the reader; details can be obtained in published texts (Farin, 1997 and Hoschek and Lasser, 1993) Bezier curves, which exist in Cartesian space, are defined parametrically. That is, the Cartesian x-, y- and z-components of a Bezier curve are each defined as functions of a parameter 't.'

A Bezier curve is fully defined by simply specifying a number points, say 'n+1,' where the curve has degree 'n.' Thus, two points yield a linear Bezier curve which is simply a straight line segment. The parameter 't' can be viewed as the fraction of distance between the two points. As 't' varies from 0 to 1, a linear line segment is traced out, representing the linear Bezier. A cubic Bezier curve is defined by four points as shown in Figure 1. These points are shown as $\mathbf{b}^0_j$ for j = 0, …, 3 and are called Bezier points or control points. Straight line segments are drawn between each pair of points $\mathbf{b}^0_j$, $\mathbf{b}^0_{j+1}$. Again, the parameter 't' can be viewed as the fraction of the distance between each pair of points on the straight line segments. The three points that are located on the straight line segments at some parameter value of 't' are given by $\mathbf{b}^1_j$. These points move along the line segments as 't' varies from 0 to 1. Additional straight line segments can be drawn between the second level of points, the $\mathbf{b}^1_j$. Again, the two points on these two line segments that are a fraction 't' from the initial points can be identified as points $\mathbf{b}^2_j$. Finally, a point on the single line segment between the points $\mathbf{b}^2_j$ at parameter value 't' is defined as point $\mathbf{b}^3_0$. Point $\mathbf{b}^3_0$ lies on the cubic Bezier curve. As 't' varies from 0 to 1, point $\mathbf{b}^3_0$ traces out a cubic curve as shown in Fig. 1.

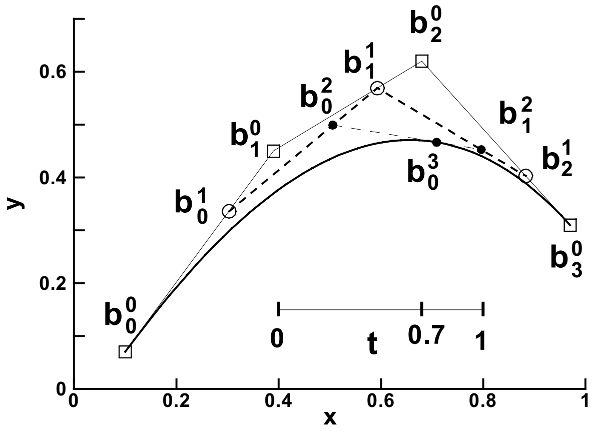

Figure 1. Construction and nomenclature of a cubic Bezier curve. The curve is traced as the parameter t varies from 0 to 1; intermediate points are shown for t = 0.7.

Bezier curves are characterized by some important properties. They are affine invariant: the Bezier points can have an affine mapping applied (such as translation or rotation) and then be evaluated or be evaluated and then mapped to yield the same curve. This is a result of the fact that Bezier curves are created by a series of linear interpolations. Bezier curves interpolate their endpoints; that is, they begin and end at the first and last Bezier points, respectively. They are very flexible: a new curve can be defined simply by making a change to any of the control points, Figure 2. Also, parametric derivatives for Bezier curves are easily found, and can be used to obtain derivatives in Cartesian space.

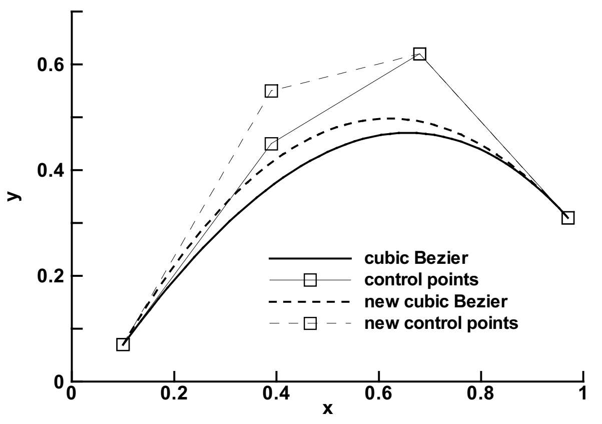

Figure 2. Modification of cubic Bezier curve by movement of a control point

When curves are needed to characterize more complex shapes, the degree of the Bezier can be increased to interpolate the shape. Alternatively, splines of Bezier curves can be created by joining individual Bezier curves together to obtain Bezier spline curves. It is usually more desirable to create spline curves as higher degree curves have a tendency to form wiggles when trying to interpolate multiple datapoints. Also, spline curves are characterized by having local support; that is, if you change a control point for one of the splines, it only affects the shape of the curve locally. A major consideration when creating spline curves is the level of continuity with which they are joined. The junction points are called knots. Without proper care, a Bezier spline curve will be only C^0 continuous at the knots; that is, the 1st derivative will not be continuous. However, if a B-spline curve rather than a Bezier spline curve is used, the spline curve will automatically be C^{n-1} continuous at the knots where n is the degree of each spline segment. A B-spline (or Basis-spline) curve is defined such that C^{n-1} continuity at the knots is preserved for n-degree B-splines curves.

B-splines were investigated as long ago as the nineteenth century by Lobachevshy, see Farin (1997). They were later used by Schoenberg (1946) for data smoothing purposes. An example of using B-splines to solve an ODE is given by de Boor (1978). B-splines can be viewed as generalizations of Bezier curves. Besides providing C^{n-1} continuity at the knots, B-splines actually require fewer data and are easier to define that are Bezier spline curves, Farin (1997). Both formulations can be used to define the exact same curve. B-splines are defined by a knot vector (a vector specifying where the junctions points are in parametric space) and the values of the associated control points or de Boor points (Farin, 1997 and Hoschek and Lasser, 1993). In general, the control points do not lie on the B-spline curve, although the end points are usually forced to end at the first and last knots (by having multiple knots at the endpoints). B-splines can be converted into Bezier spline curves following Boehm, see Farin (1997). The B-spline control points, in general, are not the same as the Bezier control points.

Figure 3 illustrates a cubic B-spline curve along with its B-spline control (de Boor) points and the data which the B-spline curve is interpolating. We use the definition of B-splines as given by Farin (1997); other authors define them slightly differently. The interpolated data come from a finite element solution (Fidap) of the axial velocity in a rectangular channel, at 1.25 channel heights downstream of the inlet. The Reynolds number for the channel flow is 500 based on channel width (=1); the inlet velocity is uniform. This location was chosen because the finite element data (bilinear elements were used) first become well-behaved at this location; that is, the transverse spatial oscillations become negligible for the finite element mesh used. The finite element solution data require 37 nodes in the transverse direction to adequately resolve the axial velocity at this location. The cubic B-spline requires the specification of values for 7 de Boor points (2 being on the channel walls) along with a knot vector of (0.,0.,0.,0.25,0.5,0.75,1.,1.,1.). The multiple knots at the endpoints are required to ensure that the B-spline curve begins at 0 and ends at 1. Hence, the bilinear finite element solution requires the determination of 35 interior node values; only 5 B-spline control points are required.

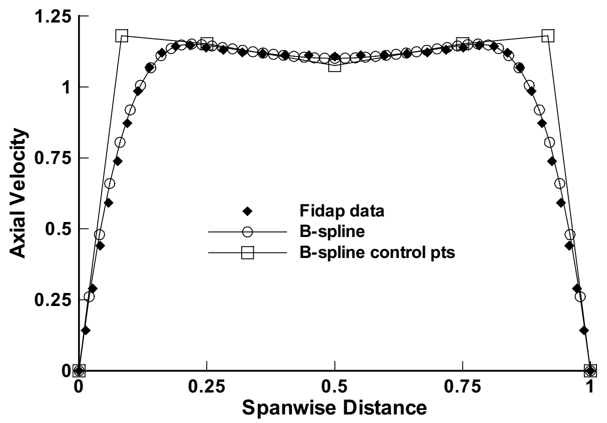

Figure 3. B-spline for the axial velocity of a channel flow at x/L=1.25 using 5 interior control points compared to finite element data (Fidap) using 35 interior grid points.

Because n-degree B-splines have C^{n-1} continuity at the knots, their derivatives are defined to degree 'n-1' everywhere. The fact that the Navier-Stokes, continuity and energy conservation equations are at most 2nd order in their derivatives, suggests the use of cubic B-spline curves. Formulae for the parametric derivatives of B-spline curves and surfaces are given in (Farin, 1997 and Hoschek and Lasser, 1993); the Cartesian derivatives can be formed from the parametric derivatives using the chain rule. The B-spline in Fig. 3 has an implicit form, meaning that the parametric coordinate is the same as the Cartesian coordinate. Figure 4 illustrates the 1st and 2nd derivatives of the B-spline curve of Fig. 3 with respect to the transverse Cartesian coordinate. The B-spline curve is first converted to a Bezier spline curve before the derivatives are computed. As can be seen, the 2nd derivative is continuous, although not smooth as expected.

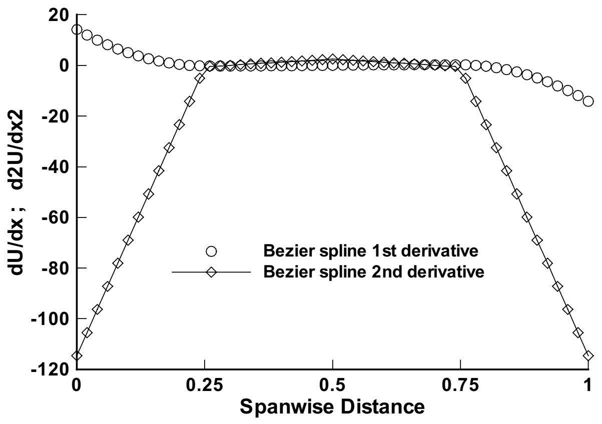

Figure 4. First and second derivatives of the axial velocity in the spanwise direction for the B-spline of Fig. 3.

Because derivatives of B-splines can be evaluated at points, all of the terms appearing in the fluid equations can be obtained at any point on a B-spline curve or surface without requiring the use of an approximation scheme such as a finite difference scheme based on Taylor series expansions. Hence, no mesh is needed for forming derivatives as required for finite differences or for performing integrations as for the finite volume and Galerkin finite element methods. There is still a requirement, however, to have a grid of knots in order to define the B-spline curve or surface. We expect, though, that the knot grid will be generated adaptively as the solution procedure progresses.

Figure 5 shows the transverse velocity at the same axial location of 1.25 channel widths. Here 14 knots or 12 de Boor points are needed to adequately resolve the velocity component profile. (The de Boor points are, in general, not located at the same points as the knots.) Of the 12 control points, 2 are on the wall, so that there are 10 data that must be found to obtain the solution for the transverse velocity component. The fact that more control points are required for the transverse velocity relative to the axial velocity indicates that the fineness of the mesh used to solve the standard finite element problem is controlled by the velocity component with the sharpest curvatures. Using B-splines, however, the number of de Boor points used can be different for the different velocity components.

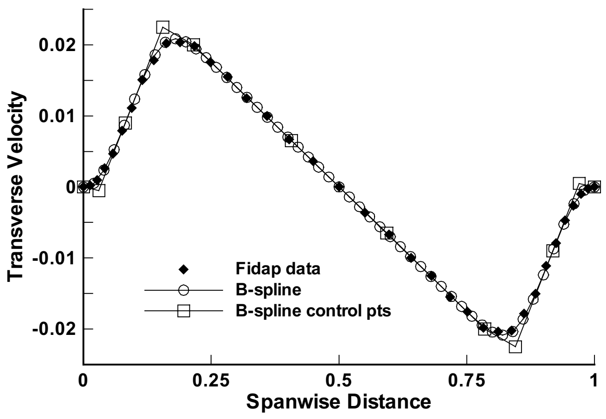

Figure 5. B-spline for transverse velocity at x/L=1.25 using 10 interior control points compared to the finite element data (Fidap) using 35 interior grid points.

Because the present method does not require a mesh in the sense of finite differences or finite volumes/elements, there are savings of computational time to be realized from being able to use only the finest knot mesh required for each dependent variable.

In summary, we can say that the proposed numerical method has the following attractive features:

1. Any level of accuracy can be achieved by using B-splines of sufficient degree with sufficient numbers of spline segments.

2. Large potential savings in solution time can be realized because fewer data are required to define solution curves and surfaces than for standard methods and each dependent variable can use the fewest data required for its solution.

3. A computational mesh in the standard sense is not required; the knot meshes that are required are expected to be generated during the solution procedure.

4. There are no truncation errors associated with the method; the errors occur in how closely the B-spline curves can represent a solution variable.

5. There are no approximate schemes needed to compute derivatives or perform quadrature.

TWO-DIMENSIONAL FLOW PROBLEM

It has been shown that B-spline curves can interpolate cross-sectional profile data from a straight channel flow using far fewer data than for a standard finite element method using bilinear elements. B-spline surfaces can be defined to represent the distribution of a variable in a two-dimensional domain. They can be formed by using a tensor product of B-spline curves, Farin (1997). An experimental code has been written that defines a B-spline surface using the tensor product of B-spline curves in a rectangular domain. The code converts the B-spline surface to a two-dimensional "patch-work" quilt of Bezier surface patches, then evaluates the surface and its 1st and 2nd derivatives in the x and y Cartesian coordinate directions. It is less computational work to convert the B-spline surface to Bezier surface patches when it is required to compute derivatives.

The B-spline code has been programmed for use in the solution of an incompressible, two-dimensional, laminar, developing flow in a straight channel at Reynolds number 500 (based on channel width). The channel flow is the same one used earlier to demonstrate the use of B-splines to interpolate cross-stream data profiles. In order to avoid the region at the entrance to the channel where the finite element solution is not adequately resolved, we use a computational domain that begins at x/L = 1.25, L being the channel width. The domain ends at x/L = 13.0. Inlet and exit profiles for the axial and transverse velocities for the B-spline computation are obtained from the finite element solution. Also, for simplicity, we use a B-spline surface interpolated to the finite element solution for the pressure for the present problem. We expect to generalize our procedure for future articles.

The B-spline surfaces for the axial (u) and transverse (v) velocity components are each defined by knot vectors in each of the two Cartesian coordinate directions and by the z-components of the de Boor (control) points. The z-components represent the dependent variables. The knot vectors are chosen to be able to interpolate the inlet data at x/L = 1.25. Figures 3 and 5 actually represent the inlet conditions for u and v. The solution is found by obtaining values for the z-components of the de Boor points. For our test problem here, this was performed by employing a commercial optimization code, OptdesX (1996). The two momentum equations and continuity equation that describe the straight channel flow are evaluated on a (47 X 50) matrix of points in the computational domain using the values and derivatives obtained by evaluating the B-spline surfaces. Initial values are set to zero except for the boundary conditions. Residuals are then computed for each equation and sent to the optimizer. The optimizer attempts to minimize the residuals by changing the control points using an optimization algorithm called the hybrid SQP-GRG (sequential quadratic programming - generalized reduced gradient) algorithm, see Parkinson and Wilson (1988). The solution procedure iterates until the residuals cannot be reduced further for a given B-spline surface. Obviously, a further reduction in the residuals could be achieved by inserting more knots, hence, more de Boor points until the desired level of accuracy is achieved.

Figures 6 and 7 illustrate (partial) B-spline solution surfaces for the axial and transverse velocities. The superimposed meshes on the surfaces show where the surfaces are evaluated for residual computation and plotting purposes; the surfaces are defined everywhere and can be evaluated anywhere. Hence, the solution surfaces actually represent bivariate geometric functions that approximate the true analytical solutions.

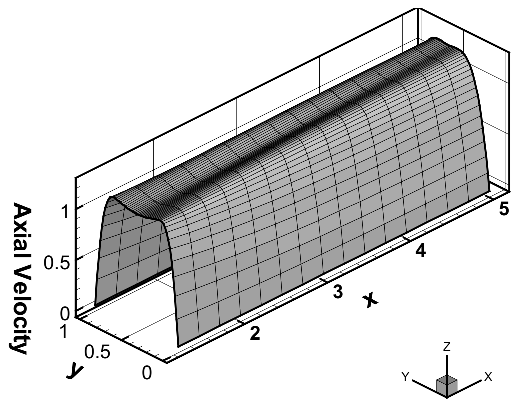

Figure 6. B-spline surface solution for the axial velocity for the 2-D channel flow.

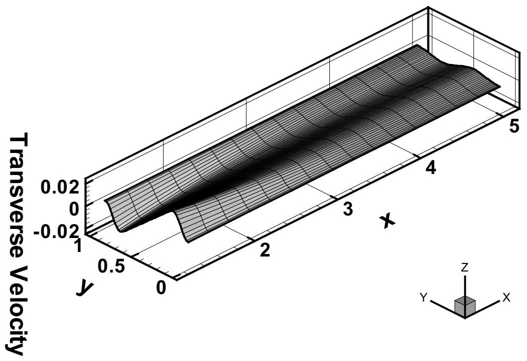

Figure 7. B-spline surface solution for the transverse velocity for the 2-D channel flow. Note that the y-direction scale is reversed from Fig. 5.

We plot the u and v velocity components at the midpoint of the computational domain in Figures 8 and 9 and compare them to the finite element solution. The finite element solution is obtained on a domain that stretches from x/L = 0 to 50 with a mesh of 100 X 37 nodes. Although the B-spline profiles do not exactly match the finite element ones, it is not clear which is closest to the exact solution. The B-spline profiles are based on the original describing differential equations, although the residuals were not reduced to the level of round-off error.

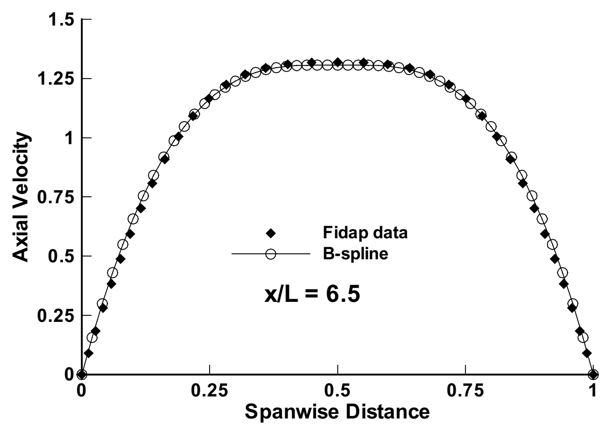

Figure 8. Comparison of axial velocity between finite element (Fidap) solution and B-spline method at x/L = 6.5.

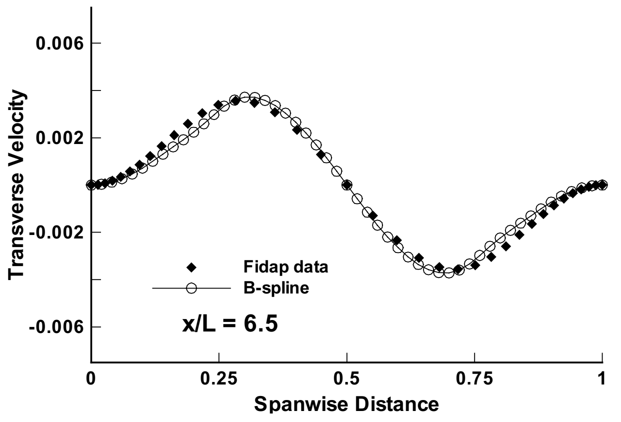

Figure 9. Comparison of transverse velocity between finite element (Fidap) solution and B-spline method at x/L = 6.5

CONCLUSIONS

We conclude that the B-spline collocation method described herein has significant potential to speed up the numerical solution of the equations of fluid mechanics due to the fact that very few data are required to approximate complex multivariate functions. We plan to investigate further procedures to solve for the de Boor points, including the use of typical solvers for standard matrix equations. We also plan to extend the method to higher dimensions and to problems that have complex geometrical domains.

ACKNOWLEDGMENTS

This work was performed at the Idaho National Engineering and Environmental Laboratory (INEEL) under the auspices of the U.S. Dept. of Energy, Idaho Operations Office, Contract No. DE-AC07-94ID13223 through the Reactive Transport in Variably Saturated Heterogeneous Media and Laboratory Directed Research and Development programs.

REFERENCES

Bloor, M.I.G., and Wilson, M.J. (1990), “Representing PDE surface in terms of B-splines,” *Computer-Aided Design,* 22, 6, 324-330.

Brown, J.M., Bloor, M.I.G., Bloor, S., and Wilson, M.J. (1990), “Generation and Modification of B-Spline Surface Approximations to PDE Surfaces Using the Finite Element Method,” in B. Ravani, ed., Advances in Design Automation, Computer Aided and Computational Design, ASME, 265-272.

Brown, J.M., Bloor, M.I.G., Bloor, S., and Wilson, M.J. (1998), “The accuracy of B-spline finite element approximations to PDE surfaces,” *Comput. Methods Appl. Mech. Engrg.,* 158, 221-234.

Carey, G.F., and Oden, J.T. (1983), *Finite Elements, A Second Course,* vol. 2, Prentice Hall Inc., Englewood Cliffs, New Jersey.

De Boor, C. (1978), *A Practical Guide to Splines,* Springer-Verlag, New York.

Farin, G. (1997), *Curves and Surfaces for Computer-Aided Geometric Design, A Practical Guide,* 4th edition, Academic Press, San Diego.

Hoschek, J., and Lasser, D. (1993), *Fundamentals of Computer Aided Geometric Design,* translated by L.L. Schumaker, A.K. Peters, Wellesley, Massachusetts.

Kravechenko, A.G., Moin, P., and Moser, R. (1996), “Zonal Embedded Grids for Numerical Simulations of Wall-Bounded Turbulent Flows,” *J. Comput. Phys.,* 127, 412-423.

Lapidus, L, and Pinder, G.F. (1982), *Numerical Solution of Partial Differential Equations, Equations in Science and Engineering,* John Wiley & Sons, New York.

OptdesX (1996), “A Software System for Optimal Engineering Design,” Design Synthesis, Inc., Provo, Utah.

Parkinson, A., and Wilson, M. (1988), “Development of a Hybrid SQP-GRG Algorithm for Constrained Nonlinear Programming,” *J. Mechan., Trans., & Automat. In Design,* Trans. of the ASME, 110, 308-315.

Rizos, D.C., and Karabalis, D.L. (1998), “A time domain BEM for 3-D elastodynamic analysis using the B-spline fundamental solutions,” *Comput. Mech.,* 22, 108-115.

Shariff, K., and Moser, R.D. (1998), “Two-Dimensional Mesh Embedding for B-spline Methods,” *J. Comput. Phys.,* 145, 471-488.