International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

Study of Cubic Spline Approximation in Solving Dynamic Economic Models

Irshad Ali[1], Dr. P. K. Chakraborty[2] , Dr. K. B. Singh[4]

[1]Research Scholar, University Department of Mathematics, B. R. A. Bihar University, Muzaffarpur, Bihar, India
[2]Department of Mathematics, M. J. K. College, Bettiah, B. R. A. Bihar University, Muzaffarpur, Bihar, India
[3]Department of Physics, L. S. College, Muzaffarpur, B. R. A. Bihar University, Muzaffarpur, Bihar, India

ARTICLEINFO

Article History:

Accepted: 10 Nov 2023
Published: 24 Nov 2023

Publication Issue
Volume 10, Issue 6
November-December-2023

Page Number
526-534

ABSTRACT

In this paper, we present about to sketch a quick background of the subject, we start with a simple model. And we present about the study of cubic spline approximation in solving Dynamic economic models.

Keywords: MATLAB, Cubic Spline, Economic Models.

1. INTRODUCTION

Given initial values of capital k_0 and technology z_0, the social planner needs to make optimal decision on the sequences of consumptions $\{c_t\}^{\infty}_{t=0}$ and capital accumulations $\{k_{t+1}\}^{\infty}_{t=0}$ throughout the time so as to [1].

$$\begin{aligned} \max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \quad & \mathcal{E}_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t)\right], \\ \text{subject to} \quad & c_t + k_{t+1} = f(k_t, z_t), \\ & z_{t+1} = \rho z_t + \epsilon_{t+1}. \end{aligned} \tag{1}$$

In the constraints, f (k_t, z_t) denotes the total production of the economy at the time period t and is determined by the current capital k_t and technology level z_t. The essence of uncertainty in this setting is caused by the evolution of the technology z_t which is assumed to follow a first order autoregressive process with normal innovation $\varrho_t \sim N(0, \sigma^2)$. In the objective function, the utility function $u(c_t)$ simulates in a broad sense the "level of satisfaction" that can be brought forth by the consumption c_t while the discount rate β characterizes the inclination of preferring consumption today than tomorrow. Both functions u and f are predetermined and their desirable properties will be described in the subsequent discussion. The operator E_0, with a noticeable subscript $_0$, emphasizes that the expected value is taken over technologies $\{z_t\}^{\infty}_{t=1}$ conditioned upon the initial technology z_0. The optimization problem (1) usually is referred to as the social planner's problem. The resulting optimal value of (1.1), as a function of the initial state of capital k_0 and technology z_0, is referred to as the value function and is denoted by $v(k_0, z_0)$.

Before continuing, we caution readers of the indicative but somewhat inadvertent misnomer that the subscript $_t$, commonly adopted in the economics literature, refers exclusively to the time period t of the economy evolution and should not be confused with the subscript $_i$ customarily used in the mathematics literature as a pointer or entry index of an array.

2. BELLMAN'S PRINCIPLE OF OPTIMALITY

Problem (1) involves infinitely many decisions simultaneously, which makes the solution extremely difficult to find. What is even more challenging is that solving (1) necessarily means to find the solution sequences $\{c_t\}^{\infty}_{t=0}$ and $\{k_{t+1}\}^{\infty}_{t=0}$ for every given initial state (k_0, z_0) because, whenever the initial state is changed, so are the subsequent decisions. One ingenious insight of great importance by Bellman reformulates the problem in a recursive form which significantly reduces the computation complexity [6]. The idea, known as Bellman's principle of optimality, asserts that an optimal policy, if exists, should have the property that the subsequent decisions from any given initial state and decision remain optimal regarding the state resulting from the first decision [3]. In other words, the value function $v(k_t, z_t)$ should satisfy the Bellman equation,

$$v(k_t, z_t) = \max_{c_t, k_{t+1}} \{u(c_t) + \beta E_t\, [v(k_{t+1}, z_{t+1})]\}\ , \qquad (2)$$

subject to the same constraints as in (1). Note that only two variables, linearly dependent due to the resource constraint, are involved in the maximization of (2).

The Bellman equation (2) typifies a general dynamic programming problem which arises in many areas other than economics [4, 6]. Quite a few numerical methods have already been proposed in the literature for solving (1,2). See, for example, a careful comparison of eight different algorithms developed from notions of either perturbation or projection in [2] and general discussions in [9, 11]. The emphasis of this paper is to propose a method that juxtapose the freedom of node collocation and the simplicity of projection without using basis. Euler equation. Though it appears often that the value function $v(k_t, z_t)$ is the underlying unknown in (2), for the purpose of decision-making it is sometimes more desirable to obtain the policy function

$$k_{t+1} = p(k_t, z_t) \qquad (3)$$

which describes the economy agent's optimal behavior with respect to state variables k_t and z_t. Repeated applications of the policy function induce the dynamics in the sequential decisions. Toward this end, we formulate the Lagrange function

$$L(c_t, k_{t+1}, z_t; \lambda_t) := u(c_t) + \beta E_t\, [v(k_{t+1}, z_{t+1})] - \lambda_t(c_t + k_{t+1} - f\,(k_t, z_t))$$

with λ_t as the multiplier. The first order optimality condition requires that

$$\begin{cases} \dfrac{\partial L}{\partial c_t} &= \dfrac{du(c_t)}{dc_t} - \lambda_t = 0, \\[2ex] \dfrac{\partial L}{\partial k_{t+1}} &= \beta \mathcal{E}_t \left[\dfrac{\partial v(k_{t+1}, z_{t+1})}{\partial k_{t+1}}\right] - \lambda_t = 0, \\[2ex] \dfrac{\partial L}{\partial \lambda_t} &= c_t + k_{t+1} - f(k_t, z_t) = 0. \end{cases} \qquad (5)$$

Applying the envelope theorem to the Bellman equation, we see that

$$\frac{\partial v(k_t, z_t)}{\partial k_t} = \frac{\partial L}{\partial k_t} = \lambda_t \frac{\partial f(k_t, z_t)}{\partial k_t}. \tag{6}$$

By eliminating the multiplier λ_t in the first two equations of (1,5), it follows that a necessary condition for optimality is

$$\frac{du(c_t)}{dc_t} = \beta \mathcal{E}_t \left[\frac{du(c_{t+1})}{dc_{t+1}} \frac{\partial f(k_{t+1}, z_{t+1})}{\partial k_{t+1}} \right], \tag{7}$$

which is known as the Euler equation for the system (2). We can replace c_t and c_{t+1} in (7) by the relationship

$$c_t = f(k_t, z_t) - k_{t+1}$$

which, after writing

$$\Gamma(k_t, k_{t+1}, z_t) := \frac{du(f(k_t, z_t) - k_{t+1})}{dc_t},$$

$$\Xi(k_t, z_t) := \frac{\partial f(k_t, z_t)}{\partial k_t},$$

leads to a 3-term finite difference equation

$$(k_t, k_{t+1}, z_t) = \beta E_t \left[(k_{t+1}, k_{t+2}; z_{t+1}) \Xi(k_{t+1}, z_{t+1}) \right] \tag{8}$$

for the unknowns $\{k_t, k_{t+1}, k_{t+2}\}$.

One point should be made clear. Starting with a given initial k_0 and an arbitrary k_1, it seems natural that the sequence $\{k_t\}$ generated by solving (8) would automatically satisfy the correspond-ing Euler equation. This is a misconception, however. Merely having a sequence of iterates satisfying (8) is not enough. The trouble is that the curve or, more precisely, the surface that interpolates these iterates may not satisfy the Euler equation in its entirety due to an incorrect value of k_1 which is supposedly equal to the evaluation of the unknown policy function $p(k_0, z_0)$, as is illustrated in Figure 1 for a fixed z_0.

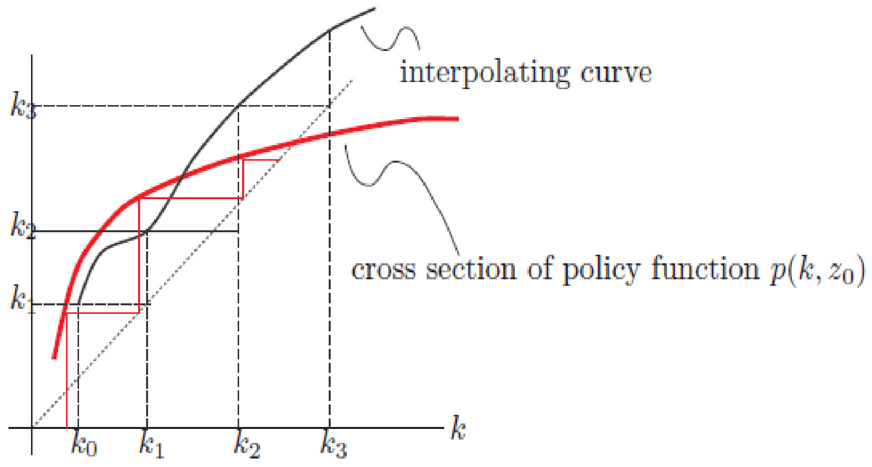

FIG. 1. Interpolating curve at discrete points obtained from (4.8) versus true policy function p (k, z_0).

What we are interested in is to find the policy function (3), which is a 2-term relationship, so as to satisfy

$$\frac{du(f(k_t,z_t)-p(k_t,z_t))}{dc_t}=\beta\mathcal{E}_t\left[\frac{du\left(f(p(k_t,z_t);z_{t+1})-p(p(k_t,z_t),z_{t+1})\right)}{dc_{t+1}}\frac{\partial f(p(k_t,z_t);z_{t+1})}{\partial k_{t+1}}\right] \quad (9)$$

for any given (k_t, z_t). Since both u and f are specified a priori, we see upon integrating the right side of (7) over z_{t+1} for the expected value that the Euler equation in general is a deterministic functional equation

$$F(k_t, z_t, p(k_t, z_t)) = 0 \qquad (10)$$

for the unknown function $p(k_t, z_t)$, where $F : R^3 \rightarrow R$ is some known nonlinear function. We shall be more specific about the constituents of the function F in the subsequent discussion, but included in the functional equation (10) for the problem (1) is the interesting but challenging task of reducing the three-term recurrence relation (4.8) to a two-term relation (3).

3. CONDITIONS FOR TERM REDUCTION

Obviously, conventional wisdom infers that the notion of reducing a three-term relationship to a two-term is impossible in general. Existence can be assumed only under some special circumstances. The following "standard" assumptions,

- $0<\beta<1$;
- The utility function u is continuously differentiable, strictly concave, and strictly increasing;
- The production function $f(\cdot, z_t)$ is continuous differentiable, concave, strictly increasing, $(0\ \cdot) \equiv 0$, and there is $k > 0$ such that

$$f(k,z_t)\begin{cases} >k & \text{if } k<\bar{k},\\ <k & \text{if } k>\bar{k};\end{cases}$$

acquired after extensive observations and explorations by economists, seem adequate to characterize general economic dynamics well. Most importantly, it is now a classical result that under these assumptions the value function and the corresponding policy function for the model problem (1) exist and are unique [6].

Of course, there are many other more complicated models taking into account other factors for economic dynamics, but the central theme is about we have just described — finding either the value function or the policy function. In this paper, our goal is to outline how the notion of 1-dimensional spline approximation can readily be applied to economic dynamics. Note that we accentuate the usage of 1-dimensional spline only of which we gain many advantages. For demonstration purposes, we shall limit our attention to the neoclassical growth model with leisure choice,

$$\begin{array}{ll}\max\limits_{\{c_t,k_{t+1},\ell_t\}_{t=0}^{\infty}} & \mathcal{E}\left[\sum_{t=0}^{\infty}\beta^t\frac{\left(c_t^{\theta}(1-\ell_t)^{1-\theta}\right)^{1-\eta}}{1-\eta}\right],\\ \text{subject to} & c_t+k_{t+1}=e^{z_t}k_t^{\alpha}\ell_t^{1-\alpha}+(1-\delta)k_t,\\ & z_{t+1}=\rho z_t+\epsilon_{t+1},\end{array} \qquad (11)$$

where $0 \leq \ell_t \leq 1$ stands for the labour supply at time t and, hence, $1 - \ell_t$ denotes the leisure. This additional variable makes (11) more complicated than (1), yet our idea remains generalizable. Our emphasis is on the simplicity and sufficiency of only a few break points needed in the spline application for high dimensional and high-resolution approximation. Additionally, we exploit the succinct programming style using tensor operations.

4. SPLINE APPROXIMATION

Many options are available for approximating the policy function p(k, z). See, for example, the discussion in [2, 11, 6]. We propose to approximate the policy function for the model (11) by a 2-dimensional cubic spline and solve the resulting system of discrete Euler equation by the Newton method. Using spline approximation and recursive methods certainly is not a new idea. A comprehensive discussion on this subject can be found in the seminal book [16]. See also [7, 12, 14, 15] for applications of shape-preserving splines to dynamical programming. Our approach stands out, however, as a special 2-dimensional spline which really can be thought of as one 1-dimensional spline "weighted" by another 1-dimensional spline. Since our formulation is essentially a 1-dimensional spline, it allows us to take advantage of the easy calculation of its derivatives analytically as we shall see in the subsequent discussion.

Recall that splines are local interpolations with controlled behavior — slope, curvature, and other degrees of differentiability — at places where two local interpolating pieces meet. We mention the cubic spline as a possible interpolant only for its ease to use. The more general concept of B-spline [8] could also be used which, in particular, offers more control over the differentiability of the spline at points where the policy function displays "kinks". Because of space limitation, we choose not to explore this generalization in this presentation.

It might be instructive to first explain how the data are structured in the MATLAB environment. We then show that, even though the analytic form of the spline might be hidden from sight, we can calculate its derivatives, especially its sensitivity to interpolants, point by point up to the machine precision. Thus, we may take the full advantage of quadratic convergence of the Newton method for computing the policy function.

Representing a cubic spline. Given $\{(x_i, y_i)\}^{n}_{i=1}$, the cubic spline q(k) that interpolates these points is a piecewise function of the form that for i = 1,. . . ,n − 1,

$$q(k) = y_i + b_i(k - x_i) + c_i(k - x_i)^2 + d_i(k - x_i)^3, \quad k \in [x_i, x_{i+1}]. \tag{12}$$

Let $x := [x_1, . . . , x_n]$ and $y := [y_1, . . . , y_n]$. With appropriate boundary constraints[2], the MATLAB command

```
A = spline(x,y);
```

creates a structure field of the form

```
A =
        form:       'pp'
        breaks:  [1xn double]
        coefs:      [(n-1)x4 double]
        pieces:  n-1
        order:      4
        dim:        1
```

where n is actually the numeric n of the length of the breaks $x_1, . . . , x_n$ and coefs is an $(n-1)\times 4$ matrix, retrievable from the command A.coefs, that stores the coefficients for the spline,

$$\texttt{A.coefs} = \begin{bmatrix} d_1 & c_1 & b_1 & y_1 \\ \vdots & \vdots & & \\ d_{n-1} & c_{n-1} & b_{n-1} & y_{n-1} \end{bmatrix}.$$

As the structure field A contains the essential information of the spline, it can be passed into the Matlab command such as

qk = ppval(A,k);

which returns the evaluation q(k) at any desirable point (or array of points) k.

Derivatives of a cubic spline. For our applications, we need to compute two kinds of derivatives of a spline. First, we need the "ordinary" derivative of q(k) evaluated at y_j . Because

$$\frac{dq}{dk} = b_i + 2c_i(k - x_i) + 3d_i(k - x_i)^2$$

over the interval $[x_i, x_{i+1}]$, we may characterize the piecewise polynomial $\frac{dq}{dk}$ by the structure field dA which has the same structure as A except that its dA.coefs is modified to

```
dA.coefs(:,1) = zeros(size(A.coefs,1),1);
dA.coefs(:,2) = 3*A.coefs(:,1);
dA.coefs(:,3) = 2*A.coefs(:,2);
dA.coefs(:,4) = A.coefs(:,3);
```

and ppval(dA,k) evaluates the derivative at any given k. Next we need the sensitivity matrix of the spline to its parameters, i.e., the partial derivatives of the spline $q(k; y_1, . . . y_n)$ with respect to each y_j , j = 1, . . . , n. While it is known that the function spline(x,y) responses nonlinearly to changes in x, we argue that its response to changes in y is easy to compute. The fact comes from the realization that the coefficients (b_i, c_i, d_i) of the various cubic polynomials in the interpolating spline are entries of the solution vector to a specific tridiagonal linear system of which the square matrix on the left side of the equation is made of entries such as $x_{i+1} - x_i$ and its powers whereas the vector on the right side is made of $y_{i+1} - y_i$ and its likes, but no powers. In other words, with fixed break points $\{x_1, . . . , x_n\}$, the spline $q(k; y_1, . . . , y_n)$ depends linearly on $\{y_1, . . . , y_n\}$ [13]. It follows that the partial derivatives are splines themselves. More specifically, for j = 1, . . . n, the partial derivative dq/dy_1 is precisely the spline that interpolates the data $\{(x, e_j)\}$, where e_j is the jth standard unit vector. Taking advantage of the vector operations in MATLAB, a simple one-line command

D = spline(x, eye(n));

where eye(n) refers to the identity matrix of size $n \times n$, effectively generates the matrix D.coefs that has $n - 1$ blocks of size $n \times 4$ where the jth row in the ith block stores the coefficients of the spline that interpolates e_j over the interval $[x_i, x_{i+1}]$. The evaluation

```
ppval(D,y)';
```

yields the $n \times n$ matrix $\left[\frac{\partial q(y_i)}{\partial y_j}\right]$

2-dimensional spline. The notion of splines can be generalized to higher dimensions. See, for example, [8, 15]. One such a generalization is the so called bicubic spline which has the advantage of being straightforward and guarantees continuity of only gradient and cross-derivative. Its second derivatives however, could be discontinuous. Instead, we propose a modified cubic spline approximation as follows.

Suppose the surface

$$z = h(x, y)$$

over the domain $\Omega \subset R^2$ is to be approximated. Let $y = [y_1, . . . , y_n]$ be a preselected set of feasible z values and define

W = spline (y, eye(n));

or

W = spline (y, [zeros (n,1), eye(n), zeros (n,1)]);

with zero end slopes clamped splines. Contained in W are n splines W_j (y), j = 1, . . . , n, satisfying $W_j(y_i) = \delta_{ij}$. For each j, choose breakpoints $x_j = [x_{j1}, \dots x_{jd_j}]$ so that $(x_{js}, y_j) \in \Omega$ for all s and j and define $h_j := [h(x_{j1}, y_j), \dots, h(x_{jd_j}, y_j)]$. Note that x_i need not be the same as x_j , nor have the same dimensionality, for different i and j. Compute the cubic spline

```
L_j = spline(x_j,f_j);
```

where `x_j` $= \mathrm{x}_j$ and `f_j` $= \mathrm{f}_j$, $j = 1, \dots, n$, and define the function

$$L_j(x) := \begin{cases} \texttt{ppval(L\_j,x)}, & \text{if } \mathfrak{l}_j \le x \le \mathfrak{u}_j, \\ 0 & \text{otherwise}, \end{cases}$$

(13)

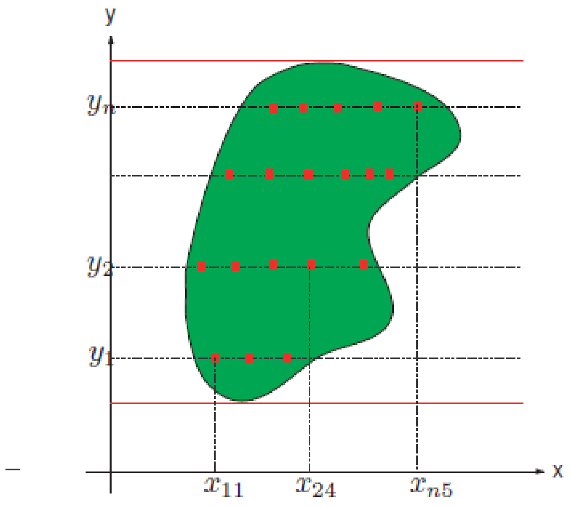

FIG. 2. Breakpoints selection over the domain Ω

where $[L_j, U_j]$ stands for the interval of cross section of the line $y = y_j$ and the domain Ω. We then define the bivariate function

$$S(x,y) := \sum_{j=1}^{n} L_j(x) W_j(y), \quad (x,y) \in \Omega.$$

(14)

It follows that for all j = 1, . . . , n and s = 1, . . . , d_j we have

$$S(x_{js}, y_j) = h(x_{js}, y_j).$$

For points that are not on the preselected grid, the functions W_j (y), J = 1, . . . , n, collectively play the role of "weighting" because for each y we have

$$\sum_{j=1}^{n} W_j(y) = 1,$$

though some of the weights might be negative. For our application, we are mainly interested in the case of rectangular domain Ω with same x_i for all i.

A **MATLAB** demonstration. Consider the peaks function

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{e^{-(x+1)^2-y^2}}{3}, \quad (x,y) \in [-3,3] \times [-3,3].$$

The following few lines of coding quickly constructs our weighted 1-D spline approximation.

```
N=20;
[x,y,z] = peaks(N);                          % generate surface test data
[xi,yi] = meshgrid(-3:.1:3,-3:.1:3);
v = peaks(xi,yi);                            % exact surface
A = spline(x(1,:),z);                        % Generate 1D splines over x
B = spline(y(:,1),[eye(N)]);                 % Generate weights over y
w = ppval(B,yi(:,1))'*ppval(A,xi(1,:));      % Weighted 1D spline
```

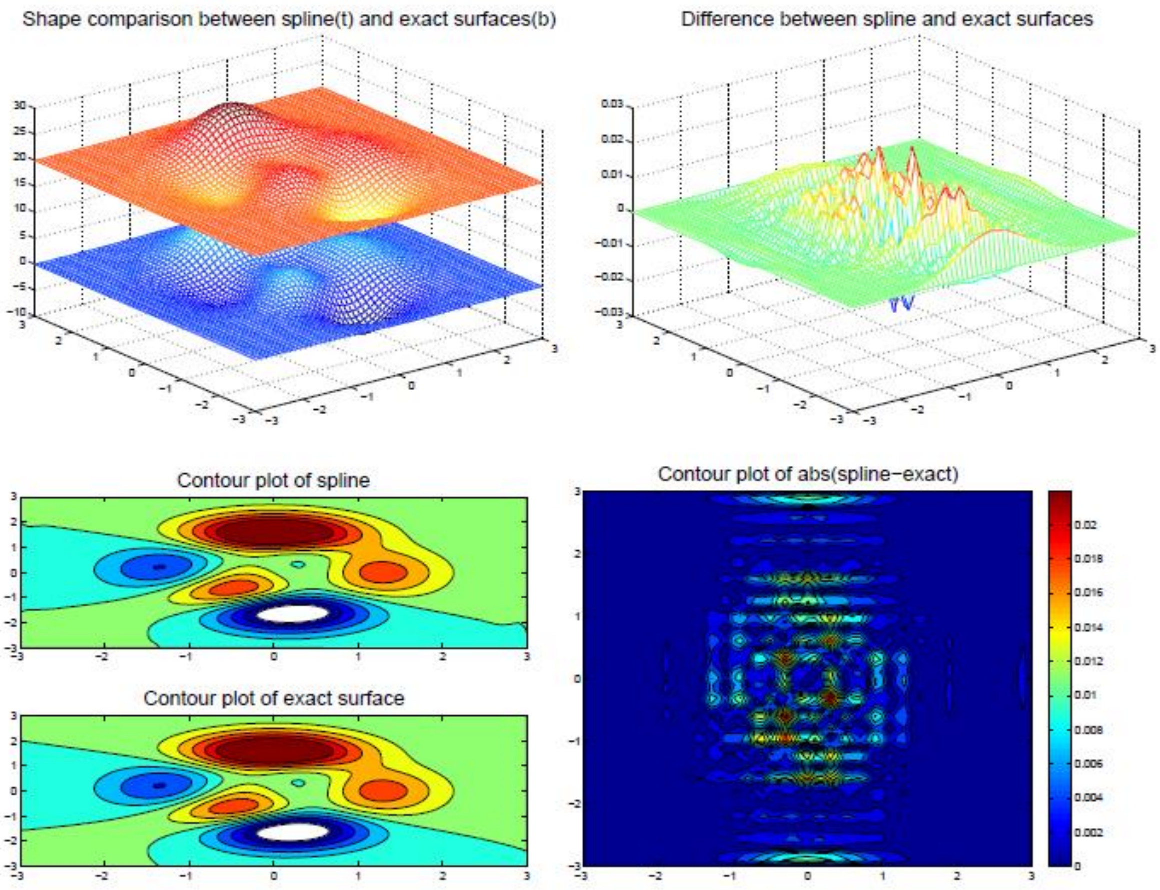

FIG. 3. Comparison between approximate surface by (14) and the exact surface.

4. CONCLUSIONS

We compare the approximate surface with the exact surface in Figure 3. In the upper left drawing, we elevate the spline surface by 20 units to show the respective terrains of the surfaces. The contour plots at the lower left drawing suggest a fairly close match of the two surfaces. The actual difference with the maximal absolute error of the order 10^{-2} is shown in the two drawings on the right column of Figure 3.

REFERENCE:

[1] H. AKIMA, A new method of interpolation and smooth curve fitting based on local procedures, J. Assoc. Comput. Mach., 17 (1970), pp. 589-602.

[2] C. DE BOOR, A Practical Guide to Splines, Springer-Verlag, New York, 1978.

[3] C. DE BOOR AND B. Swim-1-z, Piecewise monotone interpolation, J. Approximation Theory, 21 (1977), pp. 411-416.

[4] A. K. CLINE, Scalar- and planar-valued curve fitting using splines under tension, Comm. ACM, 17 (1974), pp. 218-223.

[5] It P. DUBE, Univariate blending functions and alternatives, Computer Graphics and Image Processing, 6 (1977), pp. 394-408.

[6] T. M. R. ELLIS AND D. H. McLAIN, Algorithm 514. A new method of cubic curve fitting using local data, ACM Trans. Math. Software, 3 (1977), pp. 175-178.

[7] A. R. FORREST, Curves and surfaces for computer-aided design, Ph.D. Thesis, Univ. of Cambridge, Cambridge, England, July 1968.

[8] D. F. MCALLISTER, E. PAssow AND J. A. ROULIER, Algorithms for computing shape preserving spline interpolations to data, Math. Comp., 31 (1977), pp. 717-725.

[9] D. F. MCALLISTER AND J. A. ROULIER, An algorithm for computing a shape preserving osculatory quadratic spline, ACM Trans. Math. Software, submitted.

[10] E. PASSOW, Piecewise monotone spline interpolation, J. Approximation Theory, 12 (1974), pp. 240-241.

[11] S. PRUESS, Alternatives to the exponential spline in tension, Math. Comp., to appear. [12] H. SPATH, Spline Algorithms for Curves and Surfaces, Utilitas Mathematica, Winnipeg, Canada, 1974.

[12] J. Adda and R. W. Cooper, Dynamic economics: quantitative methods and applications, The MIT Press, Cambridge, MA, 2003.

[13] S. B. Aruoba, J. Fern ́andez-Villaverde, and J. F. Rubio-Ram ́ırez, Comparing solution methods for dynamic equlibrium economies, J. Econom. Dynam. Control, 30 (2006), pp. 2477–2508.

[14] R. E. Bellman, Dynamic programming, Princeton Landmarks in Mathematics, Princeton University Press, Princeton, NJ, 2010. Reprint of the 1957 edition, With a new introduction by Stuart Dreyfus.

[15] D. P. Bertsekas, Dynamic programming and stochastic control, Academic Press, New York, 1976.

[16] C.-S. Chow and J. N. Tsitsiklis, An optimal one-way multigrid algorithm for discrete-time stochastic control, IEEE Trans. Automat. Control, 36 (1991), pp. 897–914.