http://communications.science.ankara.edu.tr

Commun.Fac.Sci.Univ.Ank.Ser. A1 Math. Stat.
Volume 74, Number 4, Pages 560–571 (2025)
https://doi.org/10.31801/cfsuasmas.1551975
ISSN 1303-5991 E-ISSN 2618-6470

Research Article; Received: September 18, 2024; Accepted: July 16, 2025

A composite method for the solution of first-order singular and nonlinear initial value problems

Nazile BUGURCAN DISIBUYUK

Department of Mathematics, Dokuz Eylül University, İzmir, TÜRKİYE

Abstract. By using the geometric and algebraic properties of Bernstein polynomials, a composite method for solving the initial value problems (IVP's) with first-order, singular and nonlinear ordinary differential equations (ODE's) has been developed. The Newton's method is incorporated into the method to solve the resulting system of nonlinear equations. The algorithm of the problem solving is reduced to the calculation of the unknown Bernstein coefficients of the approximate solution. The effectiveness of the proposed method is verified by comparing the present numerical results by other existing ones. The proposed method reduces the computation cost and gives a better approximation to the exact solution even for small degrees of approximation. Another advantage of the present method is the ability to calculate the approximate solution at each point of the solution interval in addition to the grid points.

2020 Mathematics Subject Classification. 34A45, 65L05, 65L06.
Keywords. First order singular ODE, first order nonlinear ODE, initial value problem, Bernstein polynomial, composite method.

1. Introduction

A $2D$ (or $3D$) parametric Bézier curve $P(x)$ of degree n is defined by

$$P(x) = \sum_{i=0}^{n} b_i B_i^n(x;[a,b]), \quad a \leqslant x \leqslant b,$$

where $B_i^n(x;[a,b])$ are Bernstein basis functions defined as

$$B_i^n(x;[a,b]) = \binom{n}{i} \left(\frac{x-a}{b-a}\right)^i \left(\frac{b-x}{b-a}\right)^{n-i}, \quad i = 0,1,\dots n,$$

and $\binom{n}{i}$ is the binomial coefficient, see [14]. The points $b_i \in \mathbb{R}^2$ (or $b_i \in \mathbb{R}^3$) , $i = 0,1,\dots n$, are called control points since the control polygon formed by connecting the control points gives information about the shape of the Bézier curve $P(x)$. Basically, it can be said that Bézier curves mimic the shape of their control polygons (see [9]). Thus, one may easily manipulate the shape of a Bézier curve by just changing its control points. The usefulness of Bézier curves comes from the properties of the Bernstein basis functions, $B_i^n(x;[a,b])$. For $B_i^n(x;[0,1])$, some of these properties are given as follows (see [14]):

- Partition of unity: $\sum_{i=0}^{n} B_i^n(x;[0,1]) = 1.$
- Non-negativity: $B_i^n(x;[0,1]) \geqslant 0,\ i = 0,\dots,n.$
- Symmetry: $B_{n-i}^n(x;[0,1]) = B_i^n(1-x;[0,1]),\ i = 0,\dots,n.$
- Recursion: $B_i^n(x;[0,1]) = (1-x)B_i^{n-1}(x;[0,1]) + xB_{i-1}^{n-1}(x;[0,1]),\ i = 0,\dots,n,$
  where $B_{-1}^{n-1}(x;[0,1]) = B_n^{n-1}(x;[0,1]) = 0.$

bugurcan.ruzgar@deu.edu.tr; 0000-0002-8339-1304.

- Linear Precision: $\sum_{i=0}^{n} \frac{i}{n} B_i^n(x;[0,1]) = x.$

These bases are also useful in approximation theory. Using Bernstein bases, we can also construct Bernstein polynomials which are firstly introduced by S. N. Bernstein in 1912 to give a simple, probabilistic proof of Weierstrass Approximation theorem which was originally proved in 1885 (see [6]). One of the most important results in the proof of S. N. Bernstein is given in the following theorem.

Theorem 1. *(see [6]) If $f(x)$ is a continuous real-valued function on $[0,1]$, the sequence of Bernstein polynomials*

$$B_n f(x;[0,1]) = \sum_{i=0}^{n} f\left(\frac{i}{n}\right) B_i^n(x;[0,1]) = \sum_{i=0}^{n} f\left(\frac{i}{n}\right) \binom{n}{i} x^i (1-x)^{n-i}, \quad n = 1, 2, \ldots.$$

converges uniformly to $f(x)$.

Another important property of Bernstein polynomials is about their derivatives, which can be expressed as follows:

Theorem 2. *(see [6]) If $f(x)$ is bounded in $[0,1]$ and $f^{(k)}(x_0)$ exists at a point $0 \leqslant x_0 \leqslant 1$, then* $\frac{d^k}{dx^k} B_n f(x;[0,1])\Big|_{x=x_0} \to f^{(k)}(x_0).$

For the proofs of the above theorems and more details about Bernstein polynomials see [6] and [13]. The above properties and theorems are also valid for any arbitrary interval $[a,b]$ and can be obtained simply by the transformation $x \mapsto \frac{u-a}{b-a}$ which maps the interval $[0,1]$ to the interval $[a,b]$ (see [9]).

Bézier curves and Bernstein polynomials are used widely in the literature to approximate solutions of differential equations, see [1–5,7,10,15,16,18,19]. In [3], Bernstein basis functions over the interval $[0,R]$ and matrix formulation is used to approximate solution of ordinary differential equations. [7] deals only with even-order differential equations. Higher order linear differential equations with variable coefficients are solved by using Bernstein collocation method in [4]. In [2] second-order linear differential equations with polynomial coefficients subject to Dirichlet conditions are solved by using the orthonormal relation of Bernstein basis functions with its dual basis to determine the expansion coefficients to construct a solution. The common idea for solving differential equations by using Bézier curves or Bernstein polynomials is either to use collocation method or operational matrices or least square solutions. The main problem in all of these methods is the computational cost, that is, increasing the degree of approximation increases the number of operations drastically. Conventional approximation methods utilizing the Bernstein polynomial framework often require the construction of larger matrices as the number of nodes increases, which presents significant computational challenges. The proposed method seeks to mitigate this problem by keeping matrix dimensions smaller, even with an increasing number of nodes, thus improving efficiency. [1] manages to overcome this challenge by developing a composite residual method that gives good results even in a small degree of approximation. This new composite method approximates the solution of differential equations under some constraints. These constraints are used to minimize the residual and are obtained by forcing the first few terms (depending on the degree of the differential equation) of the Taylor series of the residual to be zero. Our aim is to solve the first-order singular and nonlinear initial value problems by using Bernstein polynomials and to develop a new composite method without imposing any constraints. The proposed composite method uses only the geometric and algebraic properties of Bernstein polynomials.

The outline of the paper is as follows: In Section 2, Bernstein polynomial approximation is combined with the Newton's method to solve the initial value problems for the first-order ordinary differential equations. A test problem is used to serve numerical comparisons of the present method with others. A theorem is proved to show that the Bernstein polynomial approximation converges to the solution of the IVP. In Section 3, the proposed method of Section 2 is used to construct a composite method. The effectiveness of the proposed method is shown by using test examples and comparing present results by others. The conclusions are summarized in Section 4. The notation list used in this paper is presented in the Appendix.

2. Formulation of the Numerical Algorithm

The initial value problems for the first-order ordinary differential equations that are singular at the initial point in the form

$$y'(x) = \frac{Af_1(x, y(x))}{x-a} + f_2(x, y(x)),\ x \in [a, b],\ \begin{cases} IC_1:\ y(a) = \alpha, & A = 0 \\ IC_2:\ y(a) = \alpha,\ y'(a) = \beta, & A \neq 0 \end{cases}, \tag{1}$$

where A is a constant and the functions $f_1(x, y)$ and $f_2(x, y)$ are continuous and Lipschitz in the dependent variable y, are considered. Note that when $A = 0$, the term $\dfrac{Af_1(x, y(x))}{x-a}$ is considered to be zero and hence the IVP (1) with IC_1 has a unique solution for $A = 0$. On the other hand, if $A \neq 0$, the right hand side of the IVP does not satisfy the conditions of the existence and uniqueness theorem of first order IVPs. This is, for $A \neq 0$, the IVP (1) with the IC_1 may have a family of solutions (see Example 3.) To obtain the desired solution, one need to impose two initial conditions.

For the proposed algorithm, the interval $[a, b]$ is divided into n subintervals of the same length, $h = \frac{b-a}{n}$, where $x_i = a+ih$, $i = 0, 1, \ldots, n$, are the grid points. Then, the exact solution of IVP (1) is approximated over the interval $[a, b]$ with its Bernstein polynomial of degree n, that is,

$$y(x) \approx u(x) = B_n y(x; [a, b]) = \sum_{i=0}^{n} y_i\, B_i^n(x; [a, b]), \tag{2}$$

where

$$y_i = y(x_i) = y\left(a + i\frac{b-a}{n}\right), \quad i = 0, 1, \ldots, n.$$

Since

$$u'(x) = \frac{n}{b-a} \sum_{i=0}^{n-1} (y_{i+1} - y_i)\, B_i^{n-1}(x; [a, b]), \tag{3}$$

(see [9]), substituting $u(x)$ instead of $y(x)$ in the differential equation of IVP (1) gives

$$\begin{aligned} u' - \frac{Af_1(x, u)}{x-a} - f_2(x, u) = &\left(\frac{n}{b-a} \sum_{i=0}^{n-1} (y_{i+1} - y_i)\, B_i^{n-1}(x; [a, b])\right) \\ &- \frac{Af_1\left(x, \sum_{i=0}^{n} y_i B_i^n(x; [a, b])\right)}{x-a} \\ &- f_2\left(x, \sum_{i=0}^{n} y_i B_i^n(x; [a, b])\right) = 0. \end{aligned} \tag{4}$$

Equation (4) is well-defined at the grid points $x_j = a + \dfrac{j(b-a)}{n}$, $j = 1, 2 \ldots, n$, since the only singularity is at the initial point $x_0 = a$. Hence, evaluating Equation (4) at the grid points gives a system of n equations with n unknowns, $y_1, \ldots, y_n$:

$$\begin{aligned} g_j(y_1, \ldots, y_n) := &\left(\frac{n}{b-a} \sum_{i=0}^{n-1} (y_{i+1} - y_i)\, B_i^{n-1}(x_j; [a, b])\right) \\ &- \frac{Af_1\left(x_j, \sum_{i=0}^{n} y_i B_i^n(x_j; [a, b])\right)}{x_j - a} \\ &- f_2\left(x_j, \sum_{i=0}^{n} y_i B_i^n(x_j; [a, b])\right) = 0, \quad j = 1, \ldots n. \end{aligned} \tag{5}$$

If $f_1(x, y)$ and $f_2(x, y)$ are linear functions in the variable y, then the above system of equations is linear and can be easily solved. Otherwise, the system of equations (5) is nonlinear. In that case, an approximate solution $\left[y_1^{(p)}, y_2^{(p)}, \cdots, y_n^{(p)}\right]$ of (5) will be computed by an iterative process, the Newton's method for system of equations. The upper index p denotes the number of iterations. The Newton's method is chosen because it is an iterative procedure capable of approximating the solution of system of nonlinear equations with remarkable accuracy.

Remark 1. *Suppose that a nonlinear system of equations satisfies the following assumptions:*

(i) *The nonlinear system of equations has a solution* y^*.
(ii) *The Jacobian is Lipschitz continuous near* y^*.
(iii) *The Jacobian is nonsingular at* y^*.

It is well known that, under these assumptions, if the starting point is sufficiently near to the solution, the Newton method applied to the nonlinear system of equations converges to the solution y^* *(see [11, 12, 17]).*

The solution procedure of the system of equations (5) by using Newton's method for the cases $A = 0$ and $A \neq 0$ is explained in Table 1 and Table 2, respectively.

TABLE 1. The solution procedure of (5) by using Newton's method for the case $A = 0$

For a given tolerance ϵ,

- ***Step 1:*** Set $y_0 = \alpha$ and $\left[y_1^{(0)}, \ldots, y_n^{(0)}\right]^T = [y_0, \ldots, y_0]^T$.
- ***Step 2:*** Evaluate

$$\begin{bmatrix} y_1^{(p+1)} \\ \vdots \\ y_n^{(p+1)} \end{bmatrix} = \begin{bmatrix} y_1^{(p)} \\ \vdots \\ y_n^{(p)} \end{bmatrix} - J^{-1}(y_1^{(p)}, \ldots, y_n^{(p)}) \begin{bmatrix} g_1(y_1^{(p)}, \ldots, y_n^{(p)}) \\ \vdots \\ g_n(y_1^{(p)}, \ldots, y_n^{(p)}) \end{bmatrix} \tag{6}$$

  where J is the Jacobian matrix of the vector valued function

$$\mathbf{g}(y_1, \ldots, y_n) = \begin{bmatrix} g_1(y_1, \ldots, y_n) \\ g_2(y_1, \ldots, y_n) \\ \vdots \\ g_n(y_1, \ldots, y_n) \end{bmatrix}$$

  which is constructed from (5).
- ***Step 3:*** If $\max\{|y_1^{(p+1)} - y_1^{(p)}|, \ldots, |y_n^{(p+1)} - y_n^{(p)}|\} \leqslant \epsilon$, then ***Stop***, else set $p = p + 1$ and go to ***Step 2***.

TABLE 2. The solution procedure of (5) by using Newton's method for the case $A \neq 0$

For a given tolerance ϵ,

- ***Step 1:*** Set $y_0 = \alpha,\ y_1 = y_0 + \frac{\beta(b-a)}{n}$ and $\left[y_2^{(0)}, \ldots, y_n^{(0)}\right]^T = [y_1, \ldots, y_1]^T$.
- ***Step 2:*** Evaluate

$$\begin{bmatrix} y_2^{(p+1)} \\ \vdots \\ y_n^{(p+1)} \end{bmatrix} = \begin{bmatrix} y_2^{(p)} \\ \vdots \\ y_n^{(p)} \end{bmatrix} - J^{-1}(y_2^{(p)}, \ldots, y_n^{(p)}) \begin{bmatrix} g_2(y_1, y_2^{(p)}, \ldots, y_n^{(p)}) \\ \vdots \\ g_n(y_1, y_2^{(p)}, \ldots, y_n^{(p)}) \end{bmatrix} \tag{7}$$

  where J is the Jacobian matrix of the vector valued function

$$\mathbf{g}(y_1, \ldots, y_n) = \begin{bmatrix} g_2(y_1, y_2, \ldots, y_n) \\ \vdots \\ g_n(y_1, y_2, \ldots, y_n) \end{bmatrix}$$

  which is constructed from (5).
- ***Step 3:*** If $\max\{|y_2^{(p+1)} - y_2^{(p)}|, \ldots, |y_n^{(p+1)} - y_n^{(p)}|\} \leqslant \epsilon$, then ***Stop***, else set $p = p + 1$ and go to ***Step 2***.

Using the fact given in Remark 1, the initial guess in Table 1 is chosen as $y_i = y_0,\ i = 1, 2, \ldots n$. Similarly, since the point x_1 is closer to the points $x_i,\ i = 2, \ldots n$ than the point x_0, it is reasonable to

choose the initial guess in Table 2 as $y_i = y_1,\ i = 2, 3, \ldots n$. On the other hand, setting $y_1 = y_0 + \dfrac{\beta(b-a)}{n}$ guarantees that the approximation obtained from Table 2 and the solution of IVP (1) has the same tangent at the initial point.

In Table 1, at the final step of Newton's method, say m, the approximate values $\tilde{y}_i = y_i^{(m+1)}$ of $y(x)$ at the grid points $x_i = a + \frac{i(b-a)}{n}$, $i = 1, 2, \ldots, n$, are obtained, that is,

$$\begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_n \end{bmatrix} = \begin{bmatrix} y_1^{(m+1)} \\ \vdots \\ y_n^{(m+1)} \end{bmatrix} \approx \begin{bmatrix} y(x_1) \\ \vdots \\ y(x_n) \end{bmatrix}. \tag{8}$$

Similarly, in Table 2 we obtain

$$\begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2^{(m+1)} \\ \vdots \\ y_n^{(m+1)} \end{bmatrix} \approx \begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \\ y(x_n) \end{bmatrix}. \tag{9}$$

By construction, the values $\tilde{y}_i,\ i = 1, 2, \ldots, n$, in (8) and in (9) approximate the solution of the IVP (1) at the grid points. On the other hand, the function

$$\tilde{u}(x) = \sum_{i=0}^{n} \tilde{y}_i B_i^n(x; [a, b]), \tag{10}$$

where $\tilde{y}_0 = y_0$, approximates the solution of the IVP (1) on the whole interval $[a, b]$, which is an advantage of the present method to obtain the approximate solution of IVP (1) in addition to the grid points.

In the following test example, the present method is compared with the second-order Runge-Kutta method and the fourth-order Runge-Kutta method.

Example 1. Consider the nonlinear test problem

$$y' = -y \ln y,\ y(0) = \frac{1}{2} \tag{11}$$

(see [8, Example 6.16]). The exact solution of IVP (11) is $y(x) = 2^{-e^{-x}}$. The maximum absolute error between the values of the exact solution at the grid points, $y(x_k)$, $k = 1, 2 \ldots, n$, and the approximate values $\tilde{y}_k$ obtained by the second-order Runge-Kutta method (RK2), fourth-order Runge-Kutta method (RK4) and the present method is given in Table 3 for different values of the number of subintervals, n. The value n also corresponds to the degree of the approximation (see (2)). In addition to that, the maximum absolute error between $y(x_k)$ and $\tilde{u}(x_k)$, see (10), is also presented. In our calculations, all values have been set to have precision 200, and Newton's method applied with a tolerance $\epsilon = 10^{-30}$. Table 3 shows that the function $\tilde{u}(x_k)$ gives a better approximation at the grid points with respect to the $\tilde{y}_k$ values. RK2 and RK4 approximate $y(x_k)$ better than $\tilde{y}_k$, but approximate much worse than $\tilde{u}(x_k)$ as the number of the subintervals n increases. Furthermore, Runge-Kutta methods approximate the solution of the IVP (11) at the grid points only. But the present method approximates the solution on the whole interval by using the function $\tilde{u}(x)$ in (10).

Note in Table 3 that, although $\tilde{y}_k$ values are approximate solutions at the grid points, the function $\tilde{u}(x_k)$ provides a better approximation at the grid points. Because the exact solution of IVP (1) is approximated over the interval $[a, b]$ with its Bernstein polynomial of degree n (see Equation (2)).

The following theorem shows that the Bernstein polynomial approximation (2) converges to the solution of IVP (1) and a better approximation to the exact solution is obtained as the degree of approximation n is increased.

Theorem 3. *Suppose that $f_1(x, y)$ and $f_2(x, y)$ are continuous and Lipschitz in y. If $u(x)$ is Bernstein polynomial approximation (2) for the solution $y(x)$ of the IVP (1), then $u(x) \to y(x)$ as $n \to \infty$ for all $x \in [a, b]$.*

Proof. Let $y(x)$ be the exact solution of IVP (1) and let $u(x)$ be the approximate solution obtained by the present method. Clearly, $u(a) = y(a)$. In order to show the convergence for $x \neq a$, fix $\tilde{x} \in (a, b]$ and

TABLE 3. Comparison of the present method with the second-order Runge-Kutta method (RK2) and the fourth-order Runge-Kutta method (RK4)

	RK2	RK4	Present method	
n	$\max_{1\leqslant k\leqslant n} \lvert y(x_k)-\tilde{y}_k\rvert$	$\max_{1\leqslant k\leqslant n} \lvert y(x_k)-\tilde{y}_k\rvert$	$\max_{1\leqslant k\leqslant n} \lvert y(x_k)-\tilde{y}_k\rvert$	$\max_{1\leqslant k\leqslant n} \lvert y(x_k)-\tilde{u}(x_k)\rvert$
4	7.67315×10^{-4}	2.69490×10^{-6}	7.30587883×10^{-3}	1.28053836×10^{-4}
8	1.80964×10^{-4}	1.62549×10^{-7}	2.91576912×10^{-3}	1.90502179×10^{-8}
16	4.39226×10^{-5}	9.97759×10^{-9}	1.34417364×10^{-3}	3.33560386×10^{-16}
32	1.08187×10^{-5}	6.17969×10^{-10}	6.47590863×10^{-4}	6.77032608×10^{-32}
64	2.68460×10^{-6}	3.84481×10^{-11}	3.18410240×10^{-4}	4.24734846×10^{-66}

consider

$$R(\tilde{x}) = u'(\tilde{x}) - \frac{Af_1(\tilde{x},u(\tilde{x}))}{\tilde{x}-a} - f_2(\tilde{x},u(\tilde{x})).$$

We need to show that $R(\tilde{x})$ tends to 0 as n tends to ∞. Since $y'(\tilde{x}) = \dfrac{Af_1(\tilde{x},y(\tilde{x}))}{\tilde{x}-a} + f_2(\tilde{x},y(\tilde{x}))$, we have

$$\begin{aligned} R(\tilde{x}) =& u'(\tilde{x}) - \frac{Af_1(\tilde{x},u(\tilde{x}))}{\tilde{x}-a} - f_2(\tilde{x},u(\tilde{x})) + \left(\frac{Af_1(\tilde{x},y(\tilde{x}))}{\tilde{x}-a} + f_2(\tilde{x},y(\tilde{x})) - y'(\tilde{x})\right) \\ =& (u'(\tilde{x})-y'(\tilde{x})) - \frac{A}{\tilde{x}-a}(f_1(\tilde{x},u(\tilde{x})) - f_1(\tilde{x},y(\tilde{x}))) - (f_2(\tilde{x},u(\tilde{x})) - f_2(\tilde{x},y(\tilde{x}))) \end{aligned}$$

hence

$$|R(\tilde{x})| \leqslant |u'(\tilde{x})-y'(\tilde{x})| + \frac{|A|}{\tilde{x}-a}|f_1(\tilde{x},u(\tilde{x})) - f_1(\tilde{x},y(\tilde{x}))| + |f_2(\tilde{x},u(\tilde{x})) - f_2(\tilde{x},y(\tilde{x}))|.$$

Since $f_1(x,y)$ and $f_2(x,y)$ are Lipschitz in y, we have

$$|R(\tilde{x})| \leqslant |u'(\tilde{x})-y'(\tilde{x})| + \left(\frac{|A|M_1}{\tilde{x}-a} + M_2\right)|u(\tilde{x})-y(\tilde{x})|,$$

where M_1 and M_2 are the Lipschitz constants. By Theorem 1 and Theorem 2, we have $u'(\tilde{x}) \to y'(\tilde{x})$ and $u(\tilde{x}) \to y(\tilde{x})$ as $n \to \infty$. Thus $R(\tilde{x}) \to 0$ as $n \to \infty$. □

Theorem 3 demonstrates that increasing the degree of the Bernstein polynomial approximation, n, improves the approximation to the exact solution of IVP (1). However, considering the Jacobian matrix's size of $n \times n$, it is not recommended to increase the approximation degree due to the computational cost. The following table shows the CPU time of the proposed method in Example 1. Note that all of our calculations were carried out with *Wolfram Mathematica®* software, an Intel® Core™ i7-4790 CPU @ 3.60 GHz processor and 16 GiB memory.

TABLE 4. Computational details for Example 1

n	# unknowns	C.N. of the Jacobian (in the first iteration)	# iterations	CPU time (in seconds)
4	4	1.39×10^{1}	6	1.088×10^{-2}
8	8	3.92×10^{2}	6	7.917×10^{-2}
16	16	7.26×10^{5}	6	1.237×10^{0}
32	32	4.35×10^{12}	6	2.799×10^{1}
64	64	2.37×10^{26}	6	7.219×10^{2}

In Table 4, the number of unknowns, the condition number (C.N.) of the Jacobian matrix evaluated at the first iteration of Newton's method, the number of iterations used in Newton's method, and the corresponding CPU time for Example 1 are presented. Note that the size of the Jacobian matrix is $n \times n$ and for the evaluation of the condition number of the Jacobian matrix, 2-norm is used. Table 4 shows that as the degree of the approximation increases, the condition number of the corresponding Jacobian

matrix also increases. Considering the condition number of the Jacobian matrix, computational cost, and CPU time, it is reasonable to divide the given interval into subintervals and apply the proposed method of Section 2 to each subinterval using a small degree of approximation, rather than increasing the degree of approximation. This idea of the composite method will be explained in the next section.

3. A Composite Method for Singular First-Order Differential Equations

The IVP (1) is considered to be solved with a composite method, which is formed by dividing the interval $[a, b]$ into N subintervals $[x_{jn}, x_{jn+n}],\ j = 0, 1, \ldots, N-1$ with equal length, $\frac{b-a}{N}$, such that

$$[a, b] = \bigcup_{j=0}^{N-1} [x_{jn}, x_{jn+n}],$$

and then applying the method of Section 2 on every subinterval of $[a, b]$. For each subinterval $[x_{jn}, x_{jn+n}]$, the grid points are $x_{jn+i} = x_{jn} + \frac{i(x_{jn+n}-x_{jn})}{n},\ i = 0, 1, \ldots n$, where $x_0 = a$ and $x_{nN} = b$. The n-th order Bernstein polynomial approximation to the solution of IVP (1) over the subinterval $[x_{jn}, x_{jn+n}]$, is given as

$$u_j(x) = B_n y(x; [x_{jn}, x_{jn+n}]) = \sum_{i=0}^{n} y_{jn+i}\, B_i^n(x; [x_{jn}, x_{jn+n}]), \quad j = 0, 1, \ldots, N-1, \tag{12}$$

where $y_{jn+i} = y(x_{jn+i})$. On each subinterval $[x_{jn}, x_{jn+n}],\ j = 0, 1, \ldots, N-1$, the present method of Section 2 is applied. That is, the piecewise approximation $u_j(x)$ in (12) is substituted instead of $y(x)$ in the differential equation of IVP (1). Following that, a system of equations is solved by using Newton's method.

On the first subinterval $[x_0, x_n]$, depending on the value A of the IVP (1), we apply either the algorithm in Table 1 or the algorithm in Table 2. That is, the number of unknowns and the initial guess for the Newton's method depend on the value A. On the other subintervals, $[x_{jn}, x_{jn+n}],\ j = 1, \ldots, N-1$, we apply algorithm in the Table 1 with the initial guess is chosen as the last approximate value found by the Newton's method on the previous interval (see (14)). The algorithm of the composite method is explained in Table 5.

In the composite method, it is recommended to choose large N values and small n values, which means that the interval $[a, b]$ in IVP (1) is aimed to be divided into more subintervals (N) and to be used small degree of Bernstein polynomial approximation (n) which produces $n \times n$ Jacobian matrix in the Newton's method. Working with small n values for each subinterval reduces the computational time compared with the proposed method in Section 2 (see Table 7). Another advantage of the composite method is that the initial guesses in the Newton's method are improved by increasing the value of N. In Example 2, the initial value problem of Example 1 is again considered to make a comparison between the composite method of [1] and the present composite method. The results show the effectiveness of the present composite method.

Example 2. Consider the IVP (11). The interval $[0, 1]$ is divided into N equally spaced subintervals and the grid points in each subinterval are chosen equally spaced which gives $x_k = \frac{k}{nN},\ k = 0, 1, \ldots, nN$. In Table 4, the maximum absolute error between the values of the exact solution at the grid points, $y(x_k),\ k = 1, 2 \ldots, n$, and the approximate values $\tilde{y}_k$ by the present composite method is shown. In addition to that, the maximum absolute error between $y(x_k)$ and $\tilde{u}(x_k)$ obtained by the present method and by the composite residual method developed in [1] are compared for different values of n and N. In the calculations all values have been set to have precision 200, and Newton's method applied with a tolerance $\epsilon = 10^{-30}$. It is seen in Table 4 that the function $\tilde{u}(x_k)$ gives a better approximation at the grid points with respect to the $\tilde{y}_k$ values. The residual method of [1] approximates $y(x_k)$ better than $\tilde{y}_k$, but approximates much worse than $\tilde{u}(x_k)$ as the number of the subintervals N increases. In Table 7, the number of unknowns, the minimum and the maximum of the number of iterations used in Newton's method and the corresponding CPU time for Example 2 are given. Note that the size of the Jacobian matrix is $n \times n$.

The order of convergence of the present method is investigated in a standard way by using the following observed orders in n and N:

TABLE 5. The algorithm of the present composite method

- ***Step 1: If*** $A = 0$, apply the algorithm in the Table 1 over the interval $[x_0, x_n]$, ***else*** apply the algorithm in the Table 2 over the interval $[x_0, x_n]$.
- ***Step 2:*** Set $j = 1$ and set
$$\tilde{u}_0(x) = \sum_{i=0}^{n} \tilde{y}_i B_i^n(x; [x_0, x_n]),$$
where $\tilde{y}_i$ are obtainded from ***Step 1***.
- ***Step 3:*** Define
$$\begin{aligned} g_k(y_{jn+1}, \ldots, y_{jn+n}) :=& \frac{n}{x_{jn+n} - x_{jn}} \sum_{i=0}^{n-1} (y_{jn+i+1} - y_{jn+i})\, B_i^{n-1}(x_{jn+k}; [x_{jn}, x_{jn+n}]) \\ & - \frac{A f_1\left(x_{jn+k}, \sum_{i=0}^{n} y_{jn+i} B_i^n(x_{jn+k}; [x_{jn}, x_{jn+n}])\right)}{x_{jn+k} - x_{jn}} \\ & - f_2\left(x_{jn+k}, \sum_{i=0}^{n} y_{jn+i} B_i^n(x_{jn+k}; [x_{jn}, x_{jn+n}])\right) = 0, \end{aligned} \tag{13}$$
for $k = 1, 2, \ldots, n$.
- ***Step 4:*** Find the approximate solution $[\tilde{y}_{jn+1}, \ldots, \tilde{y}_{jn+n}]^T$ of $[y_{jn+1}, \ldots, y_{jn+n}]^T$ by solving the system of equations obtained in ***Step 3*** with the Newton's method with a given tolerance and the initial guess
$$\left[y_{jn+1}^{(0)}, \ldots, y_{jn+n}^{(0)}\right]^T = [\tilde{y}_{jn}, \ldots, \tilde{y}_{jn}]^T. \tag{14}$$
- ***Step 5:*** Set
$$\tilde{u}_j(x) = \sum_{i=0}^{n} \tilde{y}_{jn+i} B_i^n(x; [x_{jn}, x_{jn+n}]),$$
where $\tilde{y}_{jn+i} = y_{jn+i}^{(m)}$ is the approximate value of $y(x)$ at the grid points of the $j-$th subinterval calculated at the final step of the Newton's method, say m.
- ***Step 6:*** If $j = N - 1$, then ***Stop***, else set $j = j + 1$, and go to ***Step 3***.

TABLE 6. Comparison of the present composite method with residual method of [1]

	Residual method of [1]	Present composite method	
$n,\ N$	$\max_{1 \leqslant k \leqslant nN} \lvert y(x_k) - u(x_k) \rvert$	$\max_{1 \leqslant k \leqslant nN} \lvert y(x_k) - \tilde{y}_k \rvert$	$\max_{1 \leqslant k \leqslant nN} \lvert y(x_k) - \tilde{u}(x_k) \rvert$
$n = 4,\ N = 4$	$9.45664474 \times 10^{-6}$	$4.21627225 \times 10^{-4}$	$3.53213916 \times 10^{-7}$
$n = 4,\ N = 16$	$3.52031534 \times 10^{-8}$	$2.62336851 \times 10^{-5}$	$1.28781083 \times 10^{-9}$
$n = 4,\ N = 64$	$1.35785716 \times 10^{-10}$	$1.63900622 \times 10^{-6}$	$4.95387202 \times 10^{-12}$
$n = 8,\ N = 4$	$2.52202370 \times 10^{-10}$	$1.79902313 \times 10^{-4}$	$1.15470563 \times 10^{-13}$
$n = 8,\ N = 16$	$1.57651669 \times 10^{-14}$	$1.12396135 \times 10^{-5}$	$1.55191608 \times 10^{-18}$
$n = 8,\ N = 64$	$3.35287353 \times 10^{-14}$	$7.02418003 \times 10^{-7}$	$2.32335909 \times 10^{-23}$
$n = 16,\ N = 4$	$5.28055377 \times 10^{-12}$	$8.39017975 \times 10^{-5}$	$1.4823513 \times 10^{-25}$
$n = 16,\ N = 16$	$8.05822076 \times 10^{-12}$	$5.24493429 \times 10^{-6}$	$3.47773253 \times 10^{-35}$
$n = 16,\ N = 64$	$9.13458198 \times 10^{-12}$	$3.27794226 \times 10^{-7}$	$8.01496348 \times 10^{-45}$

- *observed order for* $\tilde{y}$ *in* n :
$$ord_{\tilde{y}}^1(n, N) = \frac{Log\left(\frac{e_{n,N}}{e_{2n,N}}\right)}{Log(2)},$$

TABLE 7. Computational details for Example 2

n,N	# unknowns	min of # iterations	max of # iterations	CPU time (in seconds)
$n=4,\ N=4$	16	5	5	3.512×10^{-2}
$n=4,\ N=16$	64	5	5	1.253×10^{-1}
$n=4,\ N=64$	256	4	5	4.370×10^{-1}
$n=8,\ N=4$	32	5	5	2.889×10^{-1}
$n=8,\ N=16$	128	5	5	1.383×10^{0}
$n=8,\ N=64$	512	4	5	4.434×10^{0}
$n=16,\ N=4$	64	5	5	4.410×10^{0}
$n=16,\ N=16$	256	5	5	1.805×10^{1}
$n=16,\ N=64$	1024	4	5	5.910×10^{1}

- *observed order for $\tilde{y}$ in N :*

$$ord^2_{\tilde{y}}(n,N)=\frac{Log\left(\frac{e_{n,N}}{e_{n,2N}}\right)}{Log(2)},$$

- *observed order for $\tilde{u}$ in n :*

$$ord^1_{\tilde{u}}(n,N)=\frac{Log\left(\frac{\tilde{e}_{n,N}}{\tilde{e}_{2n,N}}\right)}{Log(2)},$$

- *observed order for $\tilde{u}$ in N :*

$$ord^2_{\tilde{u}}(n,N)=\frac{Log\left(\frac{\tilde{e}_{n,N}}{\tilde{e}_{n,2N}}\right)}{Log(2)},$$

where $e_{n,N}=\max\limits_{1\leqslant k\leqslant nN}|y(x_k)-\tilde{y}_k|$ is the maximum absolute error in $\tilde{y}_k$ values and $\tilde{e}_{n,N}=\max\limits_{1\leqslant k\leqslant nN}|y(x_k)-\tilde{u}(x_k)|$ is the maximum absolute error in $\tilde{u}(x_k)$ values.

TABLE 8. For Example 2, the observed orders for $\tilde{y}$ in n (left) and observed orders for $\tilde{y}$ in N (right)

$ord^1_{\tilde{y}}(n,N)$			
	$N=4$	$N=8$	$N=16$
$n=4$	1.22875	1.2242	1.22283
$n=8$	1.10044	1.09979	1.0996
$n=16$	1.04762	1.04739	1.04733

$ord^2_{\tilde{y}}(n,N)$			
	$N=4$	$N=8$	$N=16$
$n=4$	2.00592	2.00055	2.00058
$n=8$	2.00136	1.99918	2.00025
$n=16$	2.00072	1.99899	2.0002

TABLE 9. For Example 2, the observed orders for $\tilde{u}$ in n (left) and observed orders for $\tilde{u}$ in N (right)

$ord^1_{\tilde{u}}(n,N)$			
	$N=4$	$N=8$	$N=16$
$n=4$	21.5446	25.6114	29.6282
$n=8$	39.5028	47.3733	55.3087
$n=16$	84.2594	99.9864	115.927

$ord^2_{\tilde{u}}(n,N)$			
	$N=4$	$N=8$	$N=16$
$n=4$	4.06823	4.03125	4.01489
$n=8$	8.13505	8.04807	8.01913
$n=16$	16.0056	15.9835	16.0093

The observed orders for Example 2 are presented in Table 8 and Table 9. It is seen from Table 8 and Table 9 that the order of approximation by $\tilde{y}$ values is much smaller than the order of approximation by $\tilde{u}$ values. It is concluded from Table 9 that in approximating $\tilde{u}$ values, increasing the degree of the Bernstein polynomial approximation (n) is more effective than the increasing the number of subintervals (N).

However, considering the computational cost, it is recommended to work with the Bernstein polynomials of small degree.

The IVP solved in Example 1 and Example 2 is chosen without singularity in order to show the effectiveness of the present method compared by other methods. An IVP with singularity is solved in Example 3.

Example 3. Consider the following IVP with Riccati differential equation

$$y' = \frac{y}{x} + y(e^x - 1) - \frac{y^2}{x} + xe^x, \quad y(0) = 0, \quad 0 \leqslant x \leqslant 1. \tag{15}$$

The one parameter family of solutions of IVP (15) is $y(x) = \dfrac{x\left(c - e^{e^x} + e^{x+e^x}\right)}{e^{e^x} - c}$, $c \in \mathbb{R}$. For a more specific solution, we need one more condition. Suppose that, in addition to IVP (15), we have $y'(0) = 0$. That is,

$$y' = \frac{y}{x} + y(e^x - 1) - \frac{y^2}{x} + xe^x, \quad y(0) = 0,\ y'(0) = 0. \quad 0 \leqslant x \leqslant 1. \tag{16}$$

Since

$$\frac{d}{dx}\left(\frac{x\left(c - e^{e^x} + e^{x+e^x}\right)}{e^{e^x} - c}\right)\bigg|_{x=0} = \frac{c}{e - c},$$

the solution of IVP (16) is obtained from the general solution by setting $c = 0$, that is, the solution is $y(x) = xe^x - x$.

The interval $[0, 1]$ is divided into N equally spaced subintervals and the grid points in each subinterval are chosen equally spaced which gives $x_k = \frac{k}{nN}$, $k = 0, 1, \ldots, nN$. The maximum absolute errors between the exact solution and the approximate solution obtained by the present composite method for different values of n and N are presented in Table 10 for IVP (16). As it is seen in the previous examples, the function $\tilde{u}(x_k)$ gives a better approximation to the exact solution at the grid points with respect to the $\tilde{y}_k$ values. In the calculations, all values have been set to have precision 200, and Newton's method applied with a tolerance $\epsilon = 10^{-30}$.

TABLE 10. Maximum absolute errors for different values of n and N in Example 3

$n,\ N$	$\max\limits_{1\leqslant k\leqslant nN} \lvert y(x_k) - \tilde{y}_k \rvert$	$\max\limits_{1\leqslant k\leqslant nN} \lvert y(x_k) - \tilde{u}(x_k) \rvert$
$n = 4,\ N = 4$	$1.78248879 \times 10^{-2}$	$1.59193312 \times 10^{-5}$
$n = 4,\ N = 8$	$4.87498803 \times 10^{-3}$	$7.38490360 \times 10^{-7}$
$n = 4,\ N = 16$	$1.27251127 \times 10^{-3}$	$3.53911198 \times 10^{-8}$
$n = 6,\ N = 4$	$1.07524886 \times 10^{-2}$	$7.68042008 \times 10^{-9}$
$n = 6,\ N = 8$	$2.92854233 \times 10^{-3}$	$9.76674297 \times 10^{-11}$
$n = 6,\ N = 16$	$7.63725525 \times 10^{-4}$	$1.28153044 \times 10^{-12}$
$n = 8,\ N = 4$	$7.68709965 \times 10^{-3}$	$1.69775305 \times 10^{-12}$
$n = 8,\ N = 8$	$2.09227322 \times 10^{-3}$	$5.44009282 \times 10^{-15}$
$n = 8,\ N = 16$	$5.45547944 \times 10^{-4}$	$6.66133815 \times 10^{-16}$

The three examples above are elementary first-order, nonlinear or singular initial value problems that can be solved analytically. The next example considers a more complex IVP.

Example 4. Let

$$y' = \frac{y}{x} + y^2(e^x - 1) - \frac{y^3}{x} + xe^x, \quad y(0) = 0,\ y'(0) = 0, \quad 0 \leqslant x \leqslant 1. \tag{17}$$

The IVP (17) has the solution $y(x) = e^x x - x$. The numerical results that obtained from the proposed method is given in Table 11. In the calculations, all values have been set to have precision 200, and Newton's method applied with a tolerance $\epsilon = 10^{-30}$.

TABLE 11. Maximum absolute errors for different values of n and N in Example 4

$n,\ N$	$\max_{1\leqslant k\leqslant nN} \lvert y(x_k)-\tilde{y}_k\rvert$	$\max_{1\leqslant k\leqslant nN} \lvert y(x_k)-\tilde{u}(x_k)\rvert$
$n=4,\ N=4$	1.78202994×10^{-2}	1.97536111×10^{-5}
$n=4,\ N=8$	4.87479577×10^{-3}	9.08814769×10^{-7}
$n=4,\ N=16$	1.27250231×10^{-3}	4.27711100×10^{-8}
$n=6,\ N=4$	1.07524864×10^{-2}	9.75738312×10^{-9}
$n=6,\ N=8$	2.92854231×10^{-3}	1.22655774×10^{-10}
$n=6,\ N=16$	7.63725525×10^{-4}	1.60704783×10^{-12}
$n=8,\ N=4$	7.68709965×10^{-3}	2.14350759×10^{-12}
$n=8,\ N=8$	2.09227322×10^{-3}	6.77236045×10^{-15}
$n=8,\ N=16$	5.45547944×10^{-4}	4.44089210×10^{-16}

4. Conclusion

A new composite method for solving first-order singular and nonlinear initial value problems has been developed. The size of the matrices that are used in the approximation method with Bernstein polynomials increases as the number of nodes are increased. The motivation to introduce this new method is to deal with the small-sized matrices, even if the number of nodes is increased. Test problems are used to serve numerical comparisons of the present method with others. Numerical examples show the effectiveness of the present method. Even for a small degree of approximation, the proposed method provides a satisfactory approximation to the exact solution. Working with small degree of approximation is recommended in order to reduce the computational cost of the method. As a future work, the present method will be used to solve the IVP's with higher-order differential equations.

Declaration of Competing Interests There are no competing interests regarding the contents of the paper.

Acknowledgements The author would like to thank the anonymous reviewers for their helpful comments and feedback, which significantly improved the quality of the manuscript.

Appendix

Notation List

$y(x)$: the exact solution of IVP (1) on $[a,b]$
$y(x_i)=y_i$: the value of the exact solution of IVP (1) at the grid point x_i of $[a,b]$
$\tilde{y}_i=y_i^{(m+1)}$: the approximate solution of $y(x_i)=y_i$ found by the Newton's method after m iterations
$u(x)=\sum_{i=0}^{n} y_i\,B_i^n(x;[a,b])$: n-th degree Bernstein polynomial approximation of $y(x)$ on $[a,b]$
$\tilde{u}(x)=\sum_{i=0}^{n} \tilde{y}_i\,B_i^n(x;[a,b])$: the approximate solution of $y(x)$ on $[a,b]$ with known coefficients $\tilde{y}_i$
$y_k(x)$: exact solution of IVP (1) on the subinterval $[x_{k-1},x_k]$ of $[a,b]$
$y_k(x_i^k)=y_i^k$: the value of the exact solution of IVP (1) at the grid point x_i^k of $[x_{k-1},x_k]$
$\tilde{y}_i^k=y_i^{k(m+1)}$: the approximate solution of $y_k(x_i^k)=y_i^k$ found by the Newton's method after m iterations
$u_k(x)=\sum_{i=0}^{n} y_i^k\,B_i^n(x;[x_{k-1},x_k])$: n-th degree Bernstein polynomial approximation of $y_k(x)$ on $[x_{k-1},x_k]$
$\tilde{u}_k(x)=\sum_{i=0}^{n} \tilde{y}_i^k\,B_i^n(x;[x_{k-1},x_k])$: the approximate solution of $y_k(x)$ on $[x_{k-1},x_k]$ with known coefficients $\tilde{y}_i^k$

References

[1] Adıyaman, M. E., Öğer, V., A residual method using Bézier curves for singular nonlinear equations of Lane-Emden type, *Kuwait Journal of Science*, 44(4) (2017), 9–18.

[2] Ahmed, H. M., Solutions of 2nd-order linear differential equations subject to Dirichlet boundary conditions in a Bernstein polynomial basis, *Journal of the Egyptian Mathematical Society*, 22(2) (2014), 227–237. https://doi.org/10.1016/j.joems.2013.07.007.
[3] Bhatti, M. I., Bracken, P., Solutions of differential equations in a Bernstein polynomial basis, *Journal of Computational and Applied Mathematics*, 205 (2007), 272–280. https://doi.org/10.1016/j.cam.2006.05.002.
[4] Daşcıoğlu, A. A., Acar, N. I., Bernstein collocation method for solving linear differential equations, *Gazi University Journal of Science*, 26(4) (2013), 527–534.
[5] Daşcıoğlu, A. A., İşler, N., Bernstein collocation method for solving nonlinear differential equations, *Mathematical and Computational Applications*, 18(3) (2013), 293–300. https://doi.org/10.3390/mca18030293.
[6] Davis, P. J., Interpolation and Approximation, Dover Publications, 1975.
[7] Doha, E. H., Bhrawy, A. H., Saker, M. A., On the derivatives of Bernstein polynomials: an application for the solution of high even-order differential equations, *Boundary Value Problems*, 829543 (2011). https://doi.org/10.1155/2011/829543.
[8] Epperson, J. F., An introduction to Numerical Methods and Analysis, John Wiley & Sons, Inc., New Jersey, 2013.
[9] Farin, G. E., Curves and Surfaces for CAGD: A Practical Guide, Elsevier Science, 2002.
[10] Gürler, H., Yalçınbaş, S., A new algorithm for the numerical solution of the first order nonlinear differential equations with the mixed non-linear conditions by using Bernstein polynomials, *New Trends in Mathematical Sciences*, 3(4) (2015), 114–124.
[11] Kelley, C. T., Iterative Methods for Solving Linear and Nonlinear Equations, Vol. 16 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1995.
[12] Kelley, C. T., Solving Nonlinear Equations with Newton's Method, Society for Industrial Mathematics, 1987.
[13] Lorentz, G. G., Bernstein Polynomials, Chelsea Publishing Company, New York, 1986.
[14] Marsh, D., Applied Geometry for Computer Graphics and CAD, Springer-Verlag, London, 2005.
[15] Mittal, R. C., Rohila, R., A study of one dimensional nonlinear diffusion equations by Bernstein polynomial based differential quadrature method, *Journal of Mathematical Chemistry*, 55 (2017), 673–695. https://doi.org/10.1007/s10910-016-0703-y.
[16] Olukunle, O. E., Felix, M. O., Bernstein induced one step hybrid scheme for general solution of second order initial value problems, *Malaya Journal of Matematik*, 8(2) (2020), 350–355. https://doi.org/10.26637/MJM0802/0006.
[17] Ortega, J. M., Rheinboldt, W. C., Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.
[18] Tabrizidooz, H. R., Shabanpanah, K., Bernstein polynomial basis for numerical solution of boundary value problems, *Numer Algor*, 77 (2018), 211–228. https://doi.org/10.1007/s11075-017-0311-3.
[19] Zheng, J., Sederberg, T. W., Johnson, R. W., Least squares methods for solving differential equations using Bézier control points, *Applied Numerical Mathematics*, 48 (2004), 237–252. https://doi.org/10.1016/j.apnum.2002.01.001.