QSSOR and cubic non-polynomial spline method for the solution of two-point boundary value problems

A Sunarto¹, P Agarwal²,³, J V L Chew⁴, H Justine⁵ and J Sulaiman⁶

¹Department of Tadris Mathematics, IAIN Bengkulu Jl Raden Fatah, Pagar Dewa Kota Bengkulu, Indonesia
²Department of Mathematics, Anand International College of Engineering, Jaipur-303012, India
³Nonlinear Dynamics Research Center (NDRC), Ajman University, Ajman, UAE
⁴Faculty of Computing and Informatics, Universiti Malaysia Sabah Labuan International Campus, Malaysia
⁵Algonquin College of Applied Arts and Technology, Ottawa, ON, Canada
⁶Faculty of Science and Natural Resources, Universiti Malaysia Sabah, Malaysia

Email: ¹andang99@gmail.com, ⁴jackelchew93@ums.edu.my

Abstract. Two-point boundary value problems are commonly used as a numerical test in developing an efficient numerical method. Several researchers studied the application of a cubic non-polynomial spline method to solve the two-point boundary value problems. A preliminary study found that a cubic non-polynomial spline method is better than a standard finite difference method in terms of the accuracy of the solution. Therefore, this paper aims to examine the performance of a cubic non-polynomial spline method through the combination with the full-, half-, and quarter-sweep iterations. The performance was evaluated in terms of the number of iterations, the execution time and the maximum absolute error by varying the iterations from full-, half- to quarter-sweep. A successive over-relaxation iterative method was implemented to solve the large and sparse linear system. The numerical result showed that the newly derived QSSOR method, based on a cubic non-polynomial spline, performed better than the tested FSSOR and HSSOR methods.

1. Introduction

Two-point boundary value problems (2BVPs) are commonly used as a numerical test in developing a better numerical method than the existing ones. 2BVPs can be referred to as the simplest mathematical models that arise from many science, engineering, and economics occurrences. 2BVPs can be solved using one of the three standard approaches: shooting, finite differences, and projections. The shooting approach is usually used to solve the original 2BVP by reducing it to an appropriate two-point initial-boundary problem (2IBP). By defining the 2IBP and using any initial value technique, 2IBP’s solution can be obtained iteratively and eventually converges to the solution of the original 2BVP. As for the finite differences approach, the finite differences relative to a mesh defined on the solution domain replaces the differential equation’s derivatives. By replacing the differential equation derivatives with the finite-differences approximation, a system of algebraic equations problem in either linear or nonlinear case can be formed. The system of equation is then solved to get the approximate solutions for the 2BVP. In the projections approach, a simpler function such as the piecewise polynomial is used to approximate the solution of the 2BVP where both differential equations and boundary conditions...
are satisfied. Collocation or finite element techniques were usually conducted to furnish the approximation. Among these three different approaches, only the finite difference method is considered due to its powerful technique to extend until multidimensional and space-time situations [1].

Several researchers studied the application of a cubic non-polynomial spline method to solve 2BVPs. For instance, a cubic non-polynomial spline method has been proposed to solve two parameters singularly perturbed 2BVPs, which have a dual boundary layer on a uniform mesh [2]. Moreover, a cubic non-polynomial spline was applied to solve higher-order nonlinear 2BVPs, which arise from hydrodynamic and magnetohydrodynamic theory [4]. Besides that, a new type of cubic non-polynomial spline method was also developed, which is called a mid-knot cubic non-polynomial spline, and it was used to solve a system of second-order 2BVPs [5]. It was also found that a cubic non-polynomial spline method is better than a standard finite difference method in terms of the accuracy of the solution [6].

Due to the superiority of spline function approximation compared to a finite difference method and the effectiveness of the non-polynomial spline compared to the polynomial spline [7], this paper aims to examine the performance of a cubic non-polynomial spline method through the combination with the full-, half-, and quarter-sweep iterations. The non-polynomial spline function at cubic degree is used to discretize the 2BVP to obtain the approximation equation. Then, using the formulated approximation equation, a large and sparse linear system is formed. The performance of a cubic non-polynomial spline approximation to the solution of a linear system is evaluated in terms of the number of iterations, the execution time, and the maximum absolute error by varying the iterations from full-, half- to quarter-sweep. Since there are many iterative methods to solve the linear system [8-10], this paper implements a successive over-relaxation (SOR) iterative method to handle a large and sparse linear system. The next section will discuss the proposed numerical method to solve 2BVPs.

2. Numerical Method

Given that a second-order two-point boundary value problem (2BVP) with the boundary condition as follows [1],

\[ y'' + p(x)y' + q(x)y = g(x), x \in [0, L], \] #(1)
\[ y(0) = A_1, y(L) = A_2, \] #(2)

with \( A_i, i = 1, 2 \) are arbitrary constants whereas \( p(x), q(x), \) and \( g(x) \) are known functions. Since the functions \( p(x), q(x), \) and \( g(x) \) are imposed dependently to the boundary \([0, L] \), the solution for (1) cannot be determined by the random selection of the functions. For the discretization of (1), the domain \([0, L] \) is divided uniformly into a set of \( m \) subintervals. The value of \( m = 2^s, s \geq 2 \) is always positive. The size of the subinterval for quarter-sweep is then defined by

\[ \Delta x = \frac{L}{m} = 4h, \] #(3)

and the separation of subintervals can be seen in Figure 1.

![Figure 1 Separation of uniform subintervals for quarter-sweep](image)

As shown by Figure 1, the separation of subintervals has been used to facilitate many quarter-sweep iterations related to works of literature [11-15]. Next, the half- and full-sweep iterations,
respectively, can be formed; see Figure 2 and Figure 3. Please refer for more details about the development and implementation of half-sweep iterations [16-20].

Figure 2 Separation of uniform subintervals for half-sweep

Figure 3 Separation of uniform subintervals for full-sweep

Next, given that the general function of spline and cubic non-polynomial spline respectively as follows:

\[ S(x) = Q_i(x), x \in [x_i, x_{i+2}], i = 0, 4, \ldots, n, \#(4) \]

\[ Q_i(x) = a_i \cos k (x - x_i) + b_i \sin k (x - x_i) + c_i (x - x_i) + d_i, \#(5) \]

where \( a_i, b_i, c_i, \) and \( d_i, i = 0, 4, \ldots, n \) are arbitrary constants, and \( k \) represents the value of frequency in the trigonometric function. Generally, the discretization process of the non-polynomial spline function as shown in (4) can be done by assuming the value of \( y(x) \) as the exact solution and \( S_i \) as the non-polynomial spline approximation equation for \( y_i = y(x_i) \) which is obtained from all the segments of \( Q_i(x) \) that passes through the points \( (x_i, S_i) \) and \( (x_{i+4}, S_{i+4}) \) as shown in Figure 4.

Figure 4 The cubic non-polynomial spline function

A quarter-sweep technique can discretize the function in (5) to approximate (1). And then, \( y_i \) is assumed as the solution of the spline function which passes through the points \( (x_i, y_i) \) and \( (x_{i+4}, S_{i+4}) \). This yields the following functions:

\[ Q_i(x_i) = y_i, Q_i(x_{i+4}) = y_{i+4}, Q_i'(x_i) = D_i, Q_i'(x_{i+4}) = D_{i+4}, Q_i''(x_i) = S_i, Q_i''(x_{i+4}) = S_{i+4}, \#(6) \]

At the end of the discretization, all the constant variables are expressed in the form of \( y_i, y_{i+4}, D_i, D_{i+4}, S_i, \) and \( S_{i+4} \). The constants \( a_i, b_i, c_i, \) and \( d_i \) are expressed as
After a further simplification, a general approximation equation is obtained in the form of difference, the backward and forward finite differences

\[ c_i = \frac{y_{i+4} - y_i}{4h} + \frac{h}{\theta^2 \sin 4\theta} (S_{i+4} - S_i), \quad d_i = y_i + h^2 \frac{S_i}{\theta^2}, \quad \#(7) \]

where \( \theta = kh \) and \( i = 0, 1, 2, \ldots, n \).

The continuity condition \( Q^m_1(x) = Q^m(x), m = 0, 1, \) is conducted simultaneously once all the values of constants \( a_i, b_i, c_i \) and \( d_i \) are obtained so that the cubic non-polynomial spline approximation equation can be formed. The resultant approximation equation is written as

\[
- \left( \frac{1}{4h} \right) y_{i-4} + \left( \frac{1}{2h} \right) y_i - \left( \frac{1}{4h} \right) y_{i+4} + \left( \frac{h}{\theta \sin 4\theta} - \frac{h}{4\theta^2} \right) S_{i-4} + \left( 2h \frac{h}{\theta \sin 4\theta} - \frac{2h \cos 4\theta}{\theta^2} \right) S_i + \left( \frac{h}{\theta \sin 4\theta} - \frac{h}{4\theta^2} \right) S_{i+4} = 0. \quad \#(8)
\]

Equation (8) can be simplified into

\[
- \left( \frac{1}{4h} \right) y_{i-4} + \left( \frac{1}{2h} \right) y_i - \left( \frac{1}{4h} \right) y_{i+4} + \frac{h}{\theta \sin 4\theta} (S_{i-4} - q_{i-4} S_i - q_{i+4} S_i + 2h \beta S_i + h \alpha S_{i+4}) = 0, \quad \#(9)
\]

where \( \alpha = \left( \frac{1}{\theta \sin 4\theta} - \frac{1}{4\theta^2} \right), \beta = \left( \frac{1}{4\theta^2} - \frac{4\theta \cos 4\theta}{\theta \sin 4\theta} \right) \) and \( i = 0, 1, 2, \ldots, n \).

As mentioned earlier, three standard approaches are usually used to solve the 2BVPs. In this study, the finite difference approach is used. The finite difference schemes composed of the central finite difference, the backward and forward finite differences, are substituted into (9) to get the general approximation equation of cubic non-polynomial spline as follows.

\[
S_{i-4} = -p_{i-4} y'_{i-4} - q_{i-4} y_{i-4} + f_{i-4} S_i = -p_i y'_{i-4} - q_i y_i + f_i,
\]

\[
S_{i+4} = -p_{i+4} y'_{i+4} - q_{i+4} y_{i+4} + f_{i+4}, \quad \#(10)
\]

where \( y_{i-4} = \frac{y_{i+4} - y_{i-4}}{8h}, y_{i-4} = \frac{y_{i+4} - 3y_{i-4} + 3y_{i-4}}{8h} \) and \( y_{i+4} = \frac{y_{i+4} - 3y_{i+4} + 3y_{i+4}}{8h} \). After a further simplification, a general approximation equation is obtained in the form of

\[
\delta_i y_{i-4} + \phi_i y_i + \sigma_i y_{i+4} = F_i, \quad i = 0, 1, 2, \ldots, n, \quad \#(11)
\]

where

\[
\delta_i = -\frac{\mu_0}{4} + \frac{3}{8h} - \gamma q_{i-4} + \frac{\mu_0}{8h} - \gamma \frac{p_{i+4}}{8h},
\]

\[
\phi_i = \frac{\mu_0}{2} - \gamma \frac{p_{i-4}}{2h} - \varphi q_i + \gamma \frac{p_{i+4}}{2h},
\]

\[
\sigma_i = -\frac{\mu_0}{4} + \frac{3}{8h} - \gamma q_{i-4} + \frac{\mu_0}{8h} - \gamma \frac{3p_{i+4}}{8h},
\]

\[
F_i = -\delta f_{i-4} - \phi f_i - \sigma f_{i+4},
\]

\[
\gamma = \mu_1 h^2, \varphi = 2\mu_2 h^2, \mu_0 = 4\theta^2 \sin 4\theta, \mu_1 = 4\theta^2 - \sin 4\theta,
\]

and

\[
\mu_2 = \sin 4\theta - 4\theta \cos 4\theta.
\]

Based on (11), the corresponding linear system is generated in the following form of a matrix as follows:

\[
Ay = F, \quad \#(12)
\]

where each component of the matrix is represented by

\[
A = \begin{bmatrix}
\begin{bmatrix}
\phi_1 & \sigma_1 \\
\delta_2 & \sigma_2 \\
\delta_3 & \sigma_3 \\
\vdots & \vdots \\
\delta_{n-1} & \phi_{n-1} & \sigma_{n-1} \\
\end{bmatrix} & \delta_n & \phi_n \
\end{bmatrix}_{(n \times n)},
\]

and

\[
y = [y_1 \ y_2 \ y_3 \ \ldots \ y_{n-1} \ y_n]^T,
\]
\[ F = \begin{bmatrix} F_1 & -\delta_1 y_0 & F_2 & F_3 & \ldots & F_{n-1} & F_n - \sigma_n y_{n+1} \end{bmatrix}^T. \]

Next, the SOR iterative method’s derivation based on the linear system in (12) is presented in the following. Firstly, the coefficient matrix \( A \) in (12) can be considered in the following decomposition:

\[ A = D + L + U, \tag{13} \]

where \( D, L, \) and \( U \) are diagonal, lower triangular and upper triangular matrices, respectively. By imposing (13) into (12), the SOR formula can be derived as

\[ \tilde{y}^{(k+1)} = (1 - \omega)\tilde{y}^{(k)} + \omega(D + L)^{-1} \left( -U\tilde{y}^{(k)} + F \right), \tag{14} \]

Before solving the linear system using the SOR iterative method shown in (14), the optimum value of the parameter \( \omega \) needs to be found first. The optimum value of \( \omega \) can be determined using the formula as in [8]. For practice, the optimum value of \( \omega \), which ranges from \( 1 \leq \omega < 2 \), is obtained through several runs of computer programs. The rule of thumb is that the best approximate value or the smallest number of iterations possible shows the optimum value of \( \omega \). The overall developed method for solving 2BVPs (which can be abbreviated as QSSOR) is shown below.

Table 1 Algorithm of QSSOR

| Step | Description |
|------|-------------|
| i.   | Initialize \( y_1^{(0)} \leftarrow 0, \varepsilon \leftarrow 10^{-10} \), |
| ii.  | Set the value of \( \omega \), |
| iii. | Calculate \( y_1^{(k+1)} \), use \( y^{(k+1)} = (1 - \omega)\tilde{y}^{(k)} + \omega(D + L)^{-1} \left( -U\tilde{y}^{(k)} + F \right) \), |
| iv.  | Check the convergence of the 2BVPs solutions, use \( |y_1^{(k+1)} - y_1^{(k)}| \leq \varepsilon \). Go back to step (iii) as long as the convergence criterion is not satisfied. |

3. Results and Discussion

In this section, the numerical experiment has been conducted by solving the following 2BVPs:

**Example 1**

\[ y'' - 4y = 4 \cosh(1), x \in [0,1], \tag{15} \]

given that the exact solution for Eq. (15) is \( y(x) = \cosh(2x - 1) - \cosh(1) \).

**Example 2**

\[-\frac{d^2y}{dx^2} = 9 \sin(3x), x \in [0,1], \tag{16} \]

with its exact solution given by \( \sin(3x) \).

Using these two examples of 2BVPs, the developed QSSOR method’s performance is evaluated by comparing the two tested methods, FSSOR and HSSOR methods. It is important to point out that the two tested methods are developed in the numerical laboratory and used in Figure 2 and 3 as the main guidance. For the performance evaluation, three important criteria are observed: the number of iterations, the execution time and the maximum absolute errors. All results of the numerical experiment using FSSOR, HSSOR and QSSOR for both Example 1 and Example 2 are tabulated in Table 2 and Table 3, respectively.

Table 2 Comparison between FSSOR, HSSOR and QSSOR for Example 1

| M  | k    | Time | \( \varepsilon \) | FSSOR | HSSOR | QSSOR |
|----|------|------|-----------------|-------|-------|-------|
|    |      |      |                 | k     | Time  | \( \varepsilon \) | k     | Time  | \( \varepsilon \) | k     | Time  | \( \varepsilon \) |
| 128| 386  | 1.42 | \begin{align*} 9.6867e-06 \\
                   \omega = 1.9493 \end{align*} | 193   | 0.73  | \begin{align*} 3.8742e-05 \\
                        \omega = 1.8911 \end{align*} | 97    | 0.42  | \begin{align*} 1.5502e-04 \\
                           \omega = 1.7998 \end{align*} |

5
Based on the results tabulated in Table 2 and Table 3, it can be seen that QSSOR iterative method has lesser iterations number and execution time at different grid sizes in solving the two examples, compared to FSSOR and HSSOR iterative methods. When the methods’ accuracy is compared, FSSOR can obtain more accurate solutions, followed by HSSOR and QSSOR. It cannot be denied that to get the solutions of 2BVPs with a greater efficiency rate, the cost will always be the accuracy of the method used. This interesting finding leads to a further investigation on solving 2BVPs using the cubic non-polynomial spline method.

### 4. Conclusion

The cubic non-polynomial spline’s general function is successfully discretized to formulate the approximation equation for solving the 2BVPs. A numerical experiment using two examples of 2BVPs is conducted to get the schemes’ performance regarding their number of iterations, execution time and maximum absolute error. According to the numerical results, it can be concluded that the QSSOR and the cubic non-polynomial spline approximation approach is superior in terms of the number of iterations and execution time compared to HSSOR and FSSOR iterative methods at a different number of meshes. Future research will investigate the possible improvement of the cubic non-polynomial spline method.

### References

[1] UM Ascher, RM Mattheij and RD Russell. *Numerical solution of boundary value problems for ordinary differential equations*, SIAM, New York, 1994.

[2] K Phaneendra and G Mahesh. Fourth order computational method for two parameters singularly perturbed boundary value problem using non-polynomial cubic spline. *Int. J. Math. Comput. Sci.* 2019; 10(3), 261-275. DOI: 10.1504/IJCSM.2019.101099.

[3] A Chaurasia, Y Gupta and PC Srivastava. Numerical scheme based on non-polynomial spline functions for the system of second order boundary value problems arising in various engineering applications. *J. Appl. Comput. Mech.* 2020. DOI: 10.22055/jacm.2020.32435.2012.
[4] N Khalid, M Abbas and MK Iqbal. Non-polynomial quintic spline for solving fourth-order fractional boundary value problems involving product terms. *Appl. Math. Comput.* 2019; 349, 393-407. DOI: 10.1016/j.amc.2018.12.066.

[5] Q Ding and PJY Wong. Numerical treatment of a system of second-order boundary value problems via mid-knot cubic non-polynomial spline. *AIP Conference Proceedings.* 2019; 2116, 240008. DOI: 10.1063/1.5114239

[6] J Rashidinia, R Jalilian and V Kazemi. Spline methods for the solutions of hyperbolic equations. *Appl. Math. Comput.* 2007; 190(1), 882-886. DOI: 10.1016/j.amc.2007.01.082

[7] PK Srivastava. Study of differential equations with their polynomial and non-polynomial spline based approximation. *Acta Technica Corviniensis - Bulletin of Engineering.* 2014; 7(3), 139-150.

[8] D Young. Iterative methods for solving partial differential equations of elliptical type. *T. Am. Math. Soc.* 1954; 76(1), 92-111.

[9] Y Saad. Iterative Methods for Sparse Linear Systems, SIAM, New York, 2002.

[10] W Hackbusch. Iterative Solution of Large Sparse Systems of Equations, Springer, Amsterdam, 2012.

[11] M Othman and AR Abdullah. An efficient four points modified explicit group Poisson solver. *Int. J. Comput. Math.* 2000; 76(2), 203–217. DOI: 10.1080/00207160008805020

[12] MN Suardi, NZFM Radzuan and J Sulaiman. Performance analysis of quarter-sweep Gauss-Seidel iteration with cubic b-spline approach to solve two-point boundary value problems. *Adv. Sci. Lett.* 2018; 24(3), 1732-1735. DOI: 10.1166/asl.2018.11147

[13] JCV Lung and J Sulaiman. On quarter-sweep finite difference scheme for one-dimensional porous medium equations. *Int. J. Appl. Math.* 2020; 33(3), 439-450. DOI: 10.12732/ijam.v33i3.6

[14] A Sunarto, P Agarwal, J Sulaiman, JVL Chew and E Aruchunan. Iterative method for solving one-dimensional fractional mathematical physics model via quarter-sweep and PAOR. *Adv. Differ. Equ.* 2021; 1, 147. DOI: 10.1186/s13662-021-03310-2.

[15] A Sunarto, J Sulaiman and JCV Lung. Numerical solution of the time-fractional diffusion equations via quarter-sweep preconditioned Gauss-Seidel method. *Int. J. Appl. Math.* 2021; 34(1), 111-125. DOI: 10.12732/ijam.v34i1.5.

[16] AR Abdullah. The four point Explicit Decoupled Group (EDG) method: a fast Poisson solver. *Int. J. Comput. Math.* 1991; 38, 61-70.

[17] J Sulaiman, A Sunarto, MK Hasan and M Othman. Comparative analysis of the family of Newton-HSSOR iterative methods for the solution of nonlinear second-order two point boundary value problems. *Journal Kalam.* 2016; 9(2), 18-23.

[18] JVL Chew and J Sulaiman. Half-sweep Newton-Gauss-Seidel for implicit finite difference solution of 1D nonlinear porous medium equations. *Glob. J. Pure Appl. Math.* 2016; 12(3), 2745-2752.

[19] MN Suardi, NZFM Radzuan and J Sulaiman. Cubic b-spline solution of two-point boundary value problem using HSKSOR iteration. *Glob. J. Pure Appl. Math.* 2017; 13(11), 7921-7934.

[20] S Suparmin and A Saudi. Path planning in structured environment using harmonic potentials via half-sweep modified AOR method. *Adv. Sci. Lett.* 2018; 24(3), 1885-1891. DOI: 10.1166/asl.2018.11182.